FIDDLER: A VISUALIZATION PROTOTYPE INTERFACE FOR MAKING SENSE OF NEWSFEEDS

BY

JINGNING ZHANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Associate Professor Karrie Karahalios

Abstract

Our digital life is filled with hidden curation algorithms that are affecting our life. In the FeedVis study [24] with Facebook News Feed, more than half (62.5%) of the users were unaware of the curation algorithm. However, they started manipulating it after they discovered its existence. Multiple cases can also be found where people tried to understand or alter the algorithms with their experiments. Therefore, we built Fiddler, a visualization interface to help users make sense of curation algorithms through comparison. The system has a comparison view and a curation view. The former provides comparison of visual analytic data across time periods to support exploration and goal setting. The latter provides comparison of curated feeds to let users explore their personal curation. With the curation view, we conducted a user study regarding the non-friend Tweets on Twitter. In our study, a non-friend Tweet for a user refers to a Tweet created by someone the user is not following. At the time of our study, the curation algorithm on Twitter displayed all non-friend Tweets retweeted by users' friends. To study users' awareness of this algorithm, we listed the latest 200 Tweets in users' timelines and the non-friend Tweets among them side by side in the curation view. And we found that users were able to tell the non-friend Tweets were mostly Retweets from their friends, which showed their awareness of the curation algorithm. Then we studied users' satisfaction over the non-friend Tweets, by asking them to label which Tweets they wanted to see. On average, users wanted 27% of non-friend tweets in their timelines. According to users' explanations, their desired Tweets had interesting, controversial or entertaining topics. Finally, users reported they would like to curate their timeline through unfollowing Twitter connections after this study. With Fiddler, we hope to encourage users to explore curation algorithms and help researchers understand how users want to curate their feeds.

To Grandfather and Grandmother.

Acknowledgments

Firstly, I would like to express my sincere thanks to my adviser Professor Karahalios, for her guidance throughout my thesis research and graduate study. She led me into the world of visualization and patiently taught me research skills in multiple aspects, offering solutions to my questions. This work will not be possible without her support. I am very fortunate to have her as my thesis adviser and this research is the most rewarding experience in my graduate study. I also want to express my special thanks to Motahhare Eslami, who helped me with the study from the beginning as well as Amirhossein Aleyasen, Kristen Vaccaro and Ha Kyung Kong for helping me with system development, study design, content review and other aspects of my research. Besides, I am very grateful for the valuable support and feedback from my colleagues in the Social Space and Human Computer Interaction groups.

I would like to appreciate my family for their priceless love. Grandpa and grandma, you may never understand what I write here but I want to keep my thankfulness in this thesis, which could be the period of my life as a student. I dedicate it to you. It will be stored somewhere in this world and these words may last forever with you. For the past twenty-four years, your love has never left me no matter where I am, sitting beside you in our little yard or in front of a laptop at the other side of the earth. My gratitude is beyond words. To my parents, I want to thank you for being the mentors for my life. I may have never told you, but I love you from the bottom of my heart. Your love makes me what I am and now it is my turn to give you a wonderful life. Finally, to my girlfriend, Meng Zhao, I am glad that I can say thank you the second time in a thesis. I never fully understood the term "support from family" until I met you. And finally I am pleased to inform you that our life begins now.

Table of Contents

Chapter 1 Introduction 1

Chapter 2 Related Work 7

Chapter 3 Design 11

Chapter 4 Implementation 30

Chapter 5 User Study 38

Chapter 6 Conclusion 44

References 45

Appendix A Pre-Interview Survey 49

Appendix B Interview Questions 52

Appendix C Front-end APIs 55

Chapter 1

Introduction

In this thesis, we will discuss our study on users' sensemaking of curation algorithms with the focus on social networks. The previous FeedVis study[24] showed users started to manipulate curation algorithms once they were aware them. With Fiddler, users can poke and prod their feeds and compare the outcomes of their inputs to the feeds or the results of different curation algorithms. To our knowledge, no existing system has supported this type of environment for users to explore hidden algorithms by themselves.

We will start with discussing the background and contributions of the study, as well as the related works. Then we will demonstrate the design and implementation of Fiddler, a prototype interface we developed for helping users make sense of curation algorithm in newsfeeds. Finally, we will discuss a user study we conducted with Fiddler to understand user's awareness, satisfaction of curation algorithms and their influences on users' future behaviors on social network sites.

1.1 Background

Curation algorithms have been widely used throughout people's digital life, especially on social networks. Facebook News Feed is a good example of algorithmically curated feed, where an algorithm decides what is interesting to users and what to present to them [14]. The trending topics on Twitter are also determined by algorithms according to factors such as users' locations and whom they follow [8]. With their popularity, curation algorithms have been shaping users' experience with its power in prioritization, classification, association and filtering [23]. For example, Chiu et al. [18] found users will have more click rates and longer stays at pages with algorithmically personalized content in a blogging system.

However, despite the prevalence of curation algorithms, not all users are aware of their existences. The algorithm awareness study [24], conducted by Eslami et al., recruited 40 Facebook users and found 62.5% of them were not aware of the algorithm behind Facebook News Feeds. Besides, the researchers concluded it was not simple exposure to the algorithm but active engagement that led to algorithm awareness.

According to their follow-up study, more than half of the participants started engaging more and manipulating the curation algorithm. Some wanted to make certain friends appear

again by liking their posts or visiting their pages. Others began to explore the algorithm, discussing with others. Besides evidence provided in this study, multiple examples can be found where users were trying to understand curation algorithms on social networks.

Some users had particular goals to achieve. Caleb Garling [25] experimented with Facebook and managed to make the algorithm believe his post was very important. He believed the algorithm sensed his positive sentiment and decided to pass it on. Some users tried to use the word "Congratulations" more, expecting their posts to be rated higher [49], because of a rumored tweak in the algorithm [52]. In addition, there are tutorials for pushing Facebook posts to the top of News Feed [29]. One suggestion is to include trending topics, positive phrases and life events [45].

Other users explored the algorithm with their own experiments. In one experiment by Mat Honan [33], he liked every post on Facebook for 48 hours and tried to see how it affected his News Feed. Beyond the scope of social network, the credit score algorithm became the subject of Howie Rappaport [42], who tried to get maximum credit balance and plotted how the credit score changed .

In other cases, when algorithms change, some users became upset for not being informed and tried to "fix" the problem because the results were not ideal. Klout is a website which gives social influence scores for users with its algorithm [10]. It was overwhelmed by complaints when the scoring algorithm changed and lowered user's score [36]. In 2010, Twitter changed its algorithms and unintentionally eliminated "Justin Bieber" from the top trending. Then the fans tried to make "Dustin Biber" into top trending, assuming the algorithm was blocking the star's name [51].

Nevertheless, to users, most curation algorithms in newsfeeds are black boxes with technical details hidden inside. For Facebook, it only provides a high-level description of how it decides which post to show [9]. With the curation algorithm hidden, it is difficult for users to explore and make sense of them.

1.2 Contributions

With the above reasons, we designed and developed a prototype interface, named Fiddler, for users to poke and prod their feeds in order to make sense of curation algorithms. There are two views in Fiddler: comparison view and curation view.

The comparison view (See Fig. 1.1) supports user's experimentation on feeds through comparison via input and time variation. It presents a visual analytic summary of users' feeds. Users can test their hypothesis by changing their input to the curation algorithm in the social network site and then examine the output in this view. The entire interface is divided into columns, which represent time durations. User can create one or more columns to group and compare the data during their experiments. The data are presented in rows corresponding to features of the feed. For each table cell, there is a line chart showing the

trend of certain feature during a time period. At each column, user can also log their input actions to the algorithm or choose to view the content of the feed in a list.

The comparison view is aimed at helping users make sense of curation algorithms by supporting exploration and goal setting. According to the FeedVis study, participants who explored the Facebook algorithm needed a tool to save their inputs and view the results, so that they can establish associations between their actions and outcomes. Because of this need, the comparison view will encourage exploration by allowing users to record their inputs and interactively observe the outcome presented in analytic graphs. Also, FeedVis study [24] mentioned users have goals, such as making friends appear again. To encourage goal setting, this view also allows users to create goals and reference them in notes. Through the process of achieving goals, we hope users will consider the model of the algorithms and test their hypotheses about them. In this prototype interface, we provide support for the Facebook News Feed in this view.

In the curation view (See Fig. 1.2), we let users compare feeds from different curation algorithms, by presenting feeds in two lists side by side. The original list can be used as a control group, a basis for comparison. Then users can select a curation algorithm, whose results will appear in the curated list. Users can also label their desired posts in this list. With this tool, users can learn the outcome of different curation algorithms by comparing their results. With the desired feeds selected by users, this view can also help researchers to understand what is the desired curation from the user's perspective. In the prototype interface, we support comparing between all Tweets in Twitter Home timeline and the Tweets from accounts a user is not following. This comparison will be used in our user study.

Besides creating Fiddler, we also conducted a user study with its curation view about the non-friend Tweets in Twitter Home timeline (hereafter "*the timeline*"). In this study, we define user B is user A's friend if A is following B on Twitter. We use the term "non-friend Tweet" to refer to a Tweet that is not originally created by a user's friend. While there are reports that algorithmically selected "suggested" Tweets appear in one's timeline [40], we did not observe any "suggested" Tweets throughout the course of our implementation or study. As a result, in our study, the non-friend Tweets in a user's timeline are all Retweets of the user's friends.

It has been controversial whether to include non-friend Tweets in users' timelines. Some users complained they received Tweets they did not choose to see, when Twitter tried to insert non-friend Tweets into their timelines. Some of the attempts include adding suggested Tweets [32, 40], Tweets favorited by users' friends [44], and Tweets from accounts followed by users' friends [16]. When users follow an account, they choose to subscribe to the Tweets from that account [7]. By choosing whom to follow, they are able to control what to see in their timelines. However, for non-friend Tweets, users do not directly choose to see them, because they do not follow the authors of those Tweets. In addition, compared to original Tweets from friends, non-friend Tweets are more likely to be ignored by users and be less

interesting to them. Therefore, in this study, we use Fiddler's curation view and let users compare their timelines and the non-friend Tweets in there. With this comparison, we conduct the study from the following three aspects.

Firstly, we wanted to study whether users were aware of the presence of non-friend Tweets. During the study, we first asked user to explain a hypothetical scenario of non-friend Tweets. Then we used the curation view to display all Tweets and non-friend Tweets from their timelines. Without showing who made the Retweets, we asked users to guess why the non-friend Tweets appeared in their timelines. The study will reflect users' awareness of Twitter's curation algorithm, which showed all friends' Retweets, regardless of their authorship. From users' responses, we can also learn what are the cues that help them make the guess. These results tell us what information can help to improve algorithm awareness. Non-friend Tweets are more helpful in detecting unawareness compared to friend Tweets, because the users may identify the latter directly by their authors.

Secondly, we would like to learn whether users wanted to see non-friend Tweets in their Home timelines and what was their ideal curation over them. We asked users to give satisfaction ratings over their timelines before showing Fiddler. Then after showing the curation view, we invited them to indicate the Tweets they wanted to see in the curated list with the labeling function. Then we let them explain why they liked those Tweets. These findings can show users' satisfaction over the current curation algorithm on Twitter. And we hoped to find there were some common features among the Tweets they liked. These common features will help development of both curation algorithms and systems to help users have control over them. We asked users to label non-friend Tweets because we suspected non-friend Tweets tended to be less satisfying, compared to original ones.

Finally, we would like to know users' thoughts on how they might change their Twitter interactions, after using the curation view in our study. Therefore we asked our participants to discuss opinions to modify their non-friend Tweet reception and how knowledge from this study might influence their future Twitter behavior.

In summary, we ask three research questions in this study:

RQ1: How aware are users of the curation algorithm in their Twitter Home timeline? What information can help to improve user's algorithm awareness?

RQ2: How satisfied are users with the non-friend Tweets in their timeline? What are the commonalities in user's desired Tweets, if any?

RQ3: How do users report they may modify future behavior after seeing this comparison?

1.3 Figures

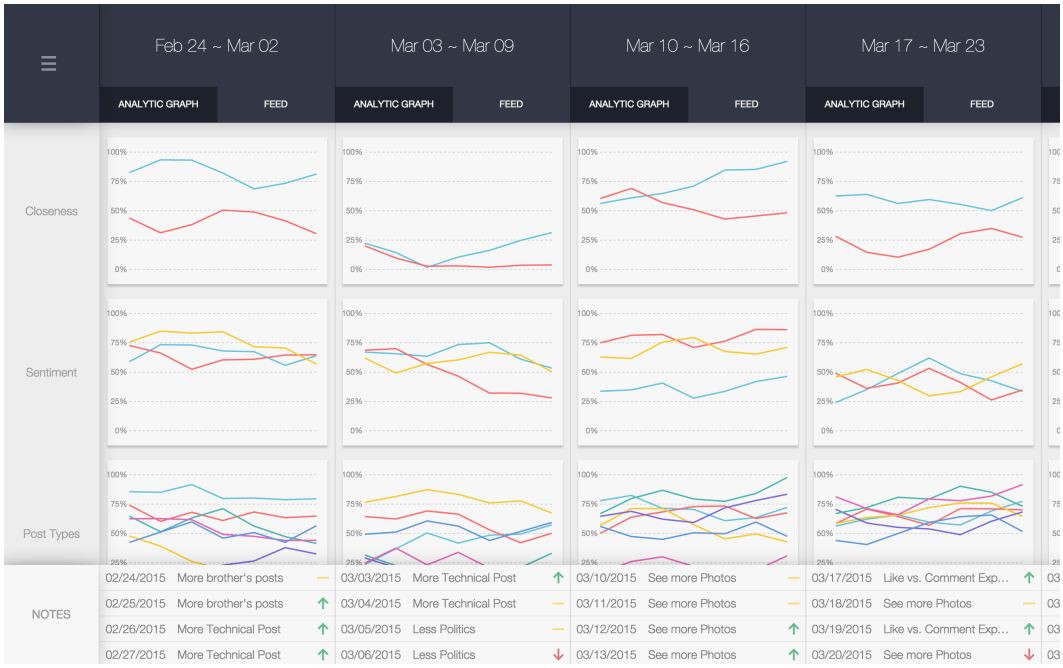


Figure 1.1: Fiddler comparison view. The structure of this view is a table. Each column contains data from a certain time period, which is customizable and is one week by default. Each row contains analytic data for one feature of the feed, such as the author's closeness to users, the average sentiment of a post and the content type of a post. There is a line chart in each table cell, showing the trends of a feature during a certain time. Each series represents one value of the feature. For instance, a graph for the "post type" feature, may contain series for "text", "photo" and "video." The bottom panel allows users to take and review notes about their actions and results. Please refer to Chapter 3 for more detail.

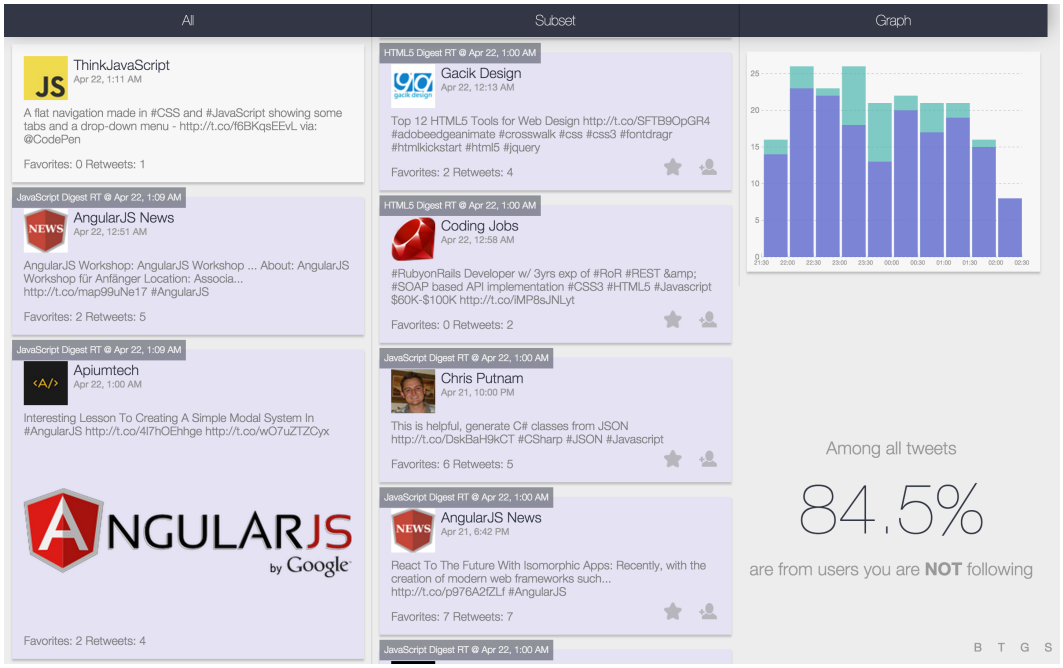


Figure 1.2: Fiddler curation view. This interface contains mainly two lists, the original and curated lists, with feeds curated by different algorithms. Users can use the original list on the left as a control feed and compare with feeds in the curated list. The histogram shows the distribution of posts over time, with two series for two lists. Feeds in the curated list have buttons for users to label whether they want to see a Tweet or follow its author. The visibility of each component could be toggled for different procedures during user study.

Chapter 2

Related Work

In this section, we will first review the preceding study of Fiddler, the FeedVis study [24] and then discuss the related literatures from two perspectives, hidden process and algorithm as well as visualization. For visualizations, we will talk about the visualization related to algorithm education, comparison and social networks.

2.1 FeedVis Study

This study, conducted by Eslami et al., was aimed at understanding user's awareness of Facebook curation algorithm and its influence on users' experience. The study was based on a system, FeedVis, which displayed the algorithmically curated and unadulterated News Feeds to users. The study invited 40 people to participate, where they used the FeedVis system and was followed-up a few months later.

This study found that 62.5% of participants were unaware of the algorithm, which did not correlate to user's usage history or size of friend network. The ignorance of the algorithm led to negative consequences, such as suspecting a remote relationship because the lack of a friend's posts. Upon discovering the algorithm, users felt betrayed yet over time satisfied with the product. They started to manipulate the algorithm after the study. Some started using the "Top stories" and "Most Recent" options to switching between different sorting options. Some set goals such as keeping friends' posts appeared in their feeds and tried to achieve them by liking friends' posts and visiting friends' personal pages. Others experimented with the algorithm and discussed with others "on ways to streamline".

Based on these findings, the researchers suggested that, comparing to explaining the algorithm's detail, providing a visual narrative of algorithmic process and enabling active engagement with the process can show users the existence of the algorithm and they were not controlled by but part of the algorithm. This was the basis of Fiddler.

2.2 Hidden Processes and Algorithms

On a wider scope, this study falls into the field of hidden processes and algorithms. For algorithm development, researcher designed online games to encapsulate difficult protein

folding problems as levels and then learned from users' explorations of the hidden folding process to create heuristics for algorithms [19]. In this project, users were not aware of the hidden process. Their goal was not to give optimized structures but to win the games. Although Fiddler supports user's exploration as well, its goal is to present the hidden curation algorithm to users through comparison, and improve user's awareness of them, instead of keeping the algorithms hidden.

About hidden algorithms, credit score is a well-known product of hidden algorithms. Credit Karma [3] provides analysis of the algorithm factors and other statistics to help users understand and manipulate the algorithm to get better score. Fiddler is similar in presenting the outcome of hidden algorithms. But compared to credit scoring, Fiddler is based on social networks, which have less financial risks for users to conduct their own experiments.

There are also researches on user's reaction toward hidden algorithms. Lavie et al. [38] studied users' opinions towards news personalization and found personalization based on general topics were preferred and the variance of interest in choices differed among topics. This conclusion is related to our findings regarding the commonalities of users' desired tweets (See RQ2). Warshaw et al. [47] examined users' reactions on algorithm-generated profiles based on their social media data. The participants found the profile creepily accurate and were faced with a paradox on whether to control the content because of privacy and the trusted impartiality of algorithms. This was an extreme case where users lost control to the algorithms. Rader et al. [41] interviewed Facebook users and found out they had different beliefs about the News Feed curation algorithm, which affected their behaviors, suggesting the feedback loop between user and the system should be better understood to avoid negative outcomes. This study was conducted with surveys on Amazon Mechanical Turk [1]. Therefore there was no way to know whether participants had seen any posts. In contrast, Fiddler is a tool that can display feeds and save users' inputs with their permission within its own interface. Our studies with Fiddler can have better monitoring over users' interactions with curation algorithms.

There are also projects trying to reverse engineering the hidden algorithms. For example, XRay [15] studied the Gmail advertisement algorithm and revealed to users which data were used as input for the algorithm. However our study is not about reverse engineering the Facebook's algorithm. There are several reasons. Firstly, the algorithm is constantly changing. Even if we succeed, our results will not be up to date when the algorithm changes. Secondly, most users do not have professional computer science background, so we cannot teach them algorithmic details. As stated above, Fiddler is aimed at encourage users to consider to model of hidden algorithms, rather than for researchers to find out their details.

2.3 Visualizations

In Fiddler, visualizations are used to summarize feed data and support users' comparison of time periods and curation algorithms. Regarding making sense of algorithms, visualizations have already been widely used for demonstration, presenting the process of algorithm execution. Algorithm visualization was regarded as a motivational factor to make students work harder [28]. Researches also showed that the way students used these visualizations had great impact on their effectiveness [35].

Comparison is a common technique used in visualizations for user to get insights from data. Gleicher et al. [27] categorized visualizations for comparison into three types of structures: juxtaposition, superposition and explicit encoding (explicitly linked the same elements in the comparison). As we can see in Fig. 1.1, both views in Fiddler use the juxtaposition structure. The authors also argued that this structure required users' memory load during comparison. We tried to eliminate this disadvantage by providing more flexibility to users regarding column operations (See Section 3.7). For visual comparisons with juxtaposition, multiple data structures have been supported. For flow data, Sambasivan et al. [46] presented server requests flow for distributed system developers so that they can understand the change of request sequences and improve performance. The visualization highlighted same elements in both alternatives. This technique is also used in our curation view by painting same tweets in different lists with the same background (See Fig. 1.2). TreeJuxtapose [39] supported hierarchical data with a main focus on scalability. Therefore it minimized the design of elements, using only lines and tags. For graph data, Semantic Graph Viewer supported both juxtaposition and superposition by providing two alternatives and a merged view.

Different from these projects, Fiddler's data are temporal data with trends of features or outcomes of curation algorithms. Its focus is not on the structures of columns but rather their contents. With comparison, we highlight the differences between durations or algorithms by placing them into different visual groups on screen. And for comparison view, this column design also allows users to group data to design their own comparisons for more insights.

Besides comparison, social network is also a popular topic for visualizations. In another categorization by Correa et al. [20], visualizations on SNS were divided into structural, temporal and analytical visualization. For structural visualizations, there were tools, such as Vizster [30], which also supported users' exploration but focused more on their friend networks. Similarly, Social Circles [43] presented friend networks in 3D to help users organize information in SNS. Regarding feeds data, Socfeedviewer [48] organized feeds according to the structures of users' friend networks. Its goal was to help users identify interesting contents in the feeds, in contrast to Fiddler's goal of making sense of the curation behind the feeds. Fiddler falls more into the category of temporal and analytical visualization. FeedWinnor [34] was also an analytic tool which helped user filtered feeds. However, it was aimed at the desired results after curation, while Fiddler helps user to compare and explore

curation algorithms themselves.

In summary, we have reviewed several works related to Fiddler from different aspects. However Fiddler is unique in that it focuses on the curation algorithms themselves rather than their output and the sensemaking of users rather than professionals. Despite the differences, there is still knowledge we can learn from these works to make Fiddler better.

Chapter 3

Design

In this section, we will discuss Fiddler system from the design perspective. We will start with the goal of designing this system and then give a walk through of Fiddler as a typical user. Then we will summarize the design challenges to achieve our goal and discuss features of Fiddler to overcome each challenge.

3.1 Goal of Fiddler

The goal of Fiddler is to provide insight into the inner workings of curation algorithm models through comparison. Fiddler is aimed to help users make sense of their newsfeeds. It contains two views: comparison view (See Fig. 1.1) and curation view (See Fig. 1.2).

The comparison view supports goal setting and exploration on curation algorithms via input and time variation. Users can conduct their own experiments to test their hypotheses of the algorithm model. After interacting with their feeds, users can review the outcome through the analytic visualizations or the text contents of their feeds. They can compare visualizations of different time periods and also save their notes and goals. For the curation view, we provide users options to compare different curated feeds and explore their personal curation. They can also label their desired posts and create their ideal feeds. Through these feeds, researchers can understand how users want to curate their feeds.

In this chapter, we will explain the features according to the design challenges for making sense of algorithms. From the design perspective, the comparison and curation view have lots of similar features. Therefore, the comparison view, being more complex, will be the main focus of this chapter. In addition, we will also introduce the challenges and features that are unique to the curation view.

3.2 Case study: comparison view

We use a hypothetical case here to demonstrate how a typical user will use the comparison view. We choose the goal-setting function as an example. The user has a goal to have more than 75% of photos among all posts in his timeline. And he had been experimenting since March 3 and this scenario happened on March 13. In the week of March 3 to 9, he liked the

first 20 photos in his feed everyday, but it seemed to have made no different. So he decided to start a second phase this week, starting March 10, to both like and comment the first 20 photos. The scenario below shows how he used the comparison view on March 13.

1. The user went to the comparison view. (Fig. 3.1a)
2. He hid the row of "Closeness" feature by clicking the hide button, so that he could focus on only the "Post type" feature. (Fig. 3.1b, Fig. 3.1c)
3. He hid the column beyond the time of his experiment to focus on the latest two weeks. (Fig. 3.1d)
4. He hovered over the graph and used the legend to hide series other than "Photo" to avoid visual clustering. (Fig. 3.1e, Fig. 3.1f)
5. From the graph in that week's column, he could see there was an increase of photos on that day. (Fig. 3.1g)
6. He compared the trend of photos in that week with the one in the previous week. He found out since he started the second phase, the photos had been increasing slowly. (Fig. 3.1h)
7. He looked at the note he wrote on March 12 when he still made little progress. And he could also tell, from the yellow result labels, his previous notes all indicated there was no big difference. (Fig. 3.1i, Fig. 3.1j)
8. Since he had observed an increase, his theory of using both comments and likes could be correct. So he took down what he did on March 13, selected the goal he was achieving and marked the result label as positive. Getting closer to the 75% goal, he though maybe he could finally have more photos on his feed as he had always wanted. (Fig. 3.1k, Fig. 3.1l, Fig. 3.1m)

3.3 Case study: curation view

Here we give another hypothetical case to show how users could benefit from the comparison of curated feeds provided by the curation view. We choose a typical Twitter user for this example. The user noticed there had been some Tweets in his Home timeline that were not from the people he followed. As we said in Section 1.2, we refer to these Tweets as non-friend Tweets in this study. The user wanted to see more original Tweets from those he followed. To curate his timeline, the user used the curation view to compare all Tweets in his timeline and all the non-friend Tweets. He performed the following steps to obtain his desired curation.

1. He opened Fiddler's curation view and chose to show all tweets in his Home timeline in the original list and non-friend Tweets in the curated list (See Fig. 3.2a).

2. He compared the two lists. The purple background highlighted the same Tweets in both lists. As he scrolled through the two lists, he found out the majority of Tweets in the original list were highlighted.

3. He also looked at the histogram and saw non-friend Tweets had taken over the majority of his timeline in the past few hours. In fact, there were nearly 60% of them in the latest 200 Tweets.

4. He looked at the Retweet information at the corner of each Tweet. He noticed there were two users who had been retweeting and filling his timeline with non-friend Tweets. And he was not interested in those Tweets.

5. He decided to curate his own timeline by unfollowing these users on Twitter.

6. He went back to the curation view (See Fig. 3.2b). Judging from the highlights, he could tell the list of all Tweets was as populated with non-friend Tweets.

7. He confirmed this result with the histogram as well as the ratio provided. 29.6% of non-friend Tweets in his timeline allowed him to see more original Tweets from his friends.

The above use case demonstrates how users could use the comparison between all and non-friend Tweets in users' timelines. This comparison is supported in the prototype interface. In the future, the curation view will support more curation algorithms at different levels for users to compare. For instance, Twitter tried to insert suggested Tweets into users' Home timeline [40]. Although we did not observe any suggested Tweet in our study, this provides a future work for the curation view. We could let users compare the Tweets in their current timelines and those suggested by Twitter to understand how users want to curate their feeds.

3.4 Challenges

With the above use cases, we want to show that Fiddler provides a straightforward way for users to observe outcome, compare results and also take notes about their input. To support sensemaking of hidden algorithms, there are several design challenges for us. Here we will first introduce the challenges for both views and then specific ones for the curation view.

3.4.1 Support comparison

Comparison of data is the key task supported by Fiddler for users to examine the outcome. Therefore, we should display the trends of different feed features for users to compare. The layout of the interface should match users' mental models, so that users could quickly find the data they are interested in.

3.4.2 Large-scale Multi-Facet Data

We must collect what users see on each day and present them on the interface. The scale of data will be very large, considering users may use it for days and weeks for their experiments. There are many features of a feed that users might be interested in. As a result, large amount of information will be presented in a limited space. The interface should support readable comparison and easy navigation to avoid overwhelming users.

3.4.3 Exploration without Restriction

Since Fiddler is designed to encourage exploration, users should not be restricted by the design of Fiddler. Firstly, we should not put arbitrary assumptions on how users will use the system. Otherwise, we may limit their creativity with our pre-determined functions. In addition, the interface should be easy to learn and use so that users will not feel frustrated and reluctant to use the interface.

3.4.4 Capturing user input

For the comparison view, we will conduct diary study in the future, which will require users to take notes for both themselves and researchers. The workload for this process should be as low as possible so that users will not feel burdened. For curation view, the labeling of Tweets should also be simple and clear to avoid any mistakes during the lab study.

3.4.5 Generalization

Fiddler will eventually support feeds from multiple platforms. As a result, the design should be easily extended for different feeds. This will help remain consistency across platforms, making it easier for users to understand the data and for researchers to extend the functions.

3.4.6 Curation View Challenges

The special challenge for curation view is the presence of large amount of feed contents on the screen. Those contents tend to be texts and photos. Users should not be distracted by other elements but focus on the comparison of curated feeds. Also, because of the large data scale, the interface should also provide to aid for comparison between lists.

With the design challenges summarized above, we now present the features in Fiddler as solutions to overcome theses challenges.

3.5 Support Comparison

3.5.1 Table Layout

To help with comparison, Fiddler has a table layout that matches users' mental models(See Fig. 1.1). The data we want to present have two critical dimensions: time and features. Therefore, we use them as the row and column definitions in the table layout, which is also a two dimensional layout. Each column contains data in a certain time period, which by default is one week and all columns are consecutive. The columns on the left are earlier. Each row contains data for one feature. In each cell of the table, there is a line chart to present the trends for a feature in a certain time. The color-coded series are possible values of the feature.

We put time as the horizontal axis in order to follow the convention that time is mostly represented horizontally. For the columns, this time division is customizable which will be discussed in Section 3.7. The reason behind the default settings is we want to present the mapping between column and period of time to users. And it will still be usable and display all data if users prefer not to customize the settings. The separation of time into columns allows users to compare graphs side by side, which contain data from different time periods. This separation can provide more possibilities for exploration than one big graph. For each graph, we choose to use line chart to show the trends of data. By default, we display the percentage axis to show the proportion of posts with a certain feature. When hovered over, the axis with absolute values will appear and also the graph's legend will show up, indicating the mapping between colors and the feature values.

The table is surround by headers on the sides. The top one shows the range of date for each column. In the left sidebar, there are labels for the feature presented in the corresponding row. These sidebars are aimed at resembling the header rows and columns in the existing spreadsheet softwares, such as Excel and Numbers. Their styles are different from the table content to stand out from other columns and rows.

Notes are located in a bar at the bottom, where notes taken within the certain time period are put in the corresponding column. Because we suggest users to take notes on the same day they perform certain activity, this design matches users input actions with their outcome in each column. For a note, only date, goal and result labels are visible by default, so that users can quickly scan and scroll through the list. Notes can be expanded to show all details when clicked. Users can choose to add, edit and delete notes through three buttons than will appear when hovered over. Since users may not manage goals on a daily basis, The goals are put in a separate page accessible from the menu on the top-left corner (See Fig. 3.3). The icons and interactions for add, delete and edit them are the same as notes to maintain consistency.

For each column, users can also see all the original contents of posts during that time period (See Fig. 3.4). We do not replicate the real Facebook style here to avoid visual clustering. For example, if a post contained multiple photos, then only the first one will be displayed. This decision will also avoid affecting the browser performance and in turn the usability.

3.5.2 Aesthetics

We choose the Material Design [11] style defined by Google, which combines both minimalism and skeuomorphism. To let the data stand out, the contents of visualization are painted with brighter and more saturated colors in contrast with the static elements in dark purple and gray. Overall we keep an analytic style with simple embellish to make it looks clear and usable [17]. In addition, aesthetic will affect users emotion and performance on the interface [22]. Therefore, we are aimed design an interface that will provide them positive emotion so that users can develop interests and use Fiddler more efficiently for a longer time [50].

3.6 Large-scale Multi-Facet Data

Large-scale data lead to a large table with many columns, rows and graphs. Our first effort is to let user easily navigate through them. The synchronized table scrolling helps to preserve useful information in header rows and columns on screen (See Fig. 3.4). As the table grows larger, users need to scroll the view to see cells that do not fit in the window size. When scrolling, the header and sidebar will scroll automatically to align with user's scrolling. The headers are frozen on the side of the screen. These headers are critical for users to locate the table cell, since the table layout is based on the values in these headers. With these frozen headers, users do not have to scroll back and forth in order to find the feature name for a row or a column's time period. The note panel is also frozen, which will reduce the efforts of locating notes and encourage user to take notes frequently.

With lots of data displayed, users may not be interested in all of them. Therefore, we allow users to hide or show a certain row or column so that they can choose which graphs to focus on (See Fig. 3.5). This function will largely reduce the number of graphs on the screen to avoid visually overwhelming users. With less graphs on screen, users will have better efficiency when locating elements [31]. Finally, users can use the "SHOW ALL" button in the menu to bring back the hidden columns and rows.

Besides the visibility of graphs, we also make the visibility of series customizable (See Fig. 3.7 and Fig. 3.8). Users can click on the corresponding item in a legend to toggle a series' visibility. The icon for a hidden series has no fill color, in order to convey the concept of "invisible". This function will help reduce the clustering within the graph.

To further reduce visual clustering, we pick line chart to use minimum space for displaying the analytic data. Line colors are arranged in a special order for series, so that they have the maximum distances with each other on the color wheel. Finally, we only display controls when they are needed. The most obvious example is the legend of graphs. Although every graph has one legend, it is impossible for users to click them all at the same time. These design ideas will help reduce the number of elements not related to the data we present.

3.7 Exploration without Restriction

Fiddler encourages user to explore the algorithm through comparison. Therefore, if we put too many design assumptions on how users will explore the data, we may restrict their creativity with the limited functions.

To encourage exploration, we allow users to add or hide columns and also to set the time period for each column (See Fig. 3.6). By clicking the edit button in the header row, users will be prompt to input a new time period for a column. Its data and notes will be updated when saved. This function breaks the restriction placed by the default settings, allowing users to compare between any time periods. Life events do not always happen periodically. Users will also have their own experiment plans. Therefore, this function provides users control over the time dimension of the overall table layout.

On the other hand, the difficulty in using the interface will discourage users. If users fail to perform operations, they may think the function is not supported or give up because of frustration. So we also create features to make the interface easy to learn and use.

We frequently use hovering to trigger operations, including the display of legends, absolute values, editing and hide buttons. Hovering makes these functions more visible. Besides, design conventions are followed throughout the system. We use the two conventional interaction modes, viewing and editing, in the forms for notes and goals. These modes appear frequently in iOS and Mac system. Regarding the frozen headers, we are mimicking the similar table interaction in Excel and Numbers. For icons, they are widely used with conventional choices. We intentionally use trashcan rather than cross icon for "delete" in order to eliminate its ambiguity with "cancel" (See Fig. 3.10). Colors are also designed to follow common senses. The result labels are color coded in the same way as stock prices, which could be helpful if a user has related experiences.

To reduce learning cost, the system contains reminders of how to use certain functions. There are descriptive place holders telling users what to write with each form field. Feedbacks are widely provided for user interactions, with all buttons having background transitions and pointer cursors. For those interactions without explicit feedbacks, messages will show up to indicate the outcome of interactions (See Fig. 3.11). In this way, users will be able to confirm their actions towards our interface.

3.8 Capturing user inputs

Although diary study has been used in many fields, asking users to take notes and create goals for us is still a burden for them. Therefore, our goal is to minimize the effort of inputing data.

We start with making users type as few free form texts as possible when writing a note (See Fig. 3.9). The date will be automatically filled with the current date and a calendar will popup for users to choose the date with one click. The goal field is a drop down list rather

than text box. It is synchronized with the goal list so that users can select a goal by clicking. The result labels are buttons for users to click and switch between positive, negative and neutral. The only text fields in this form are the active and results, which cannot be inferred. And we have reduced their sizes to give users the impression that they do not have to write long. In addition, for goal setting page, we created the exact same design with same icon and interactions. Users will not have to learn more to use the goal page.

3.9 Generalization

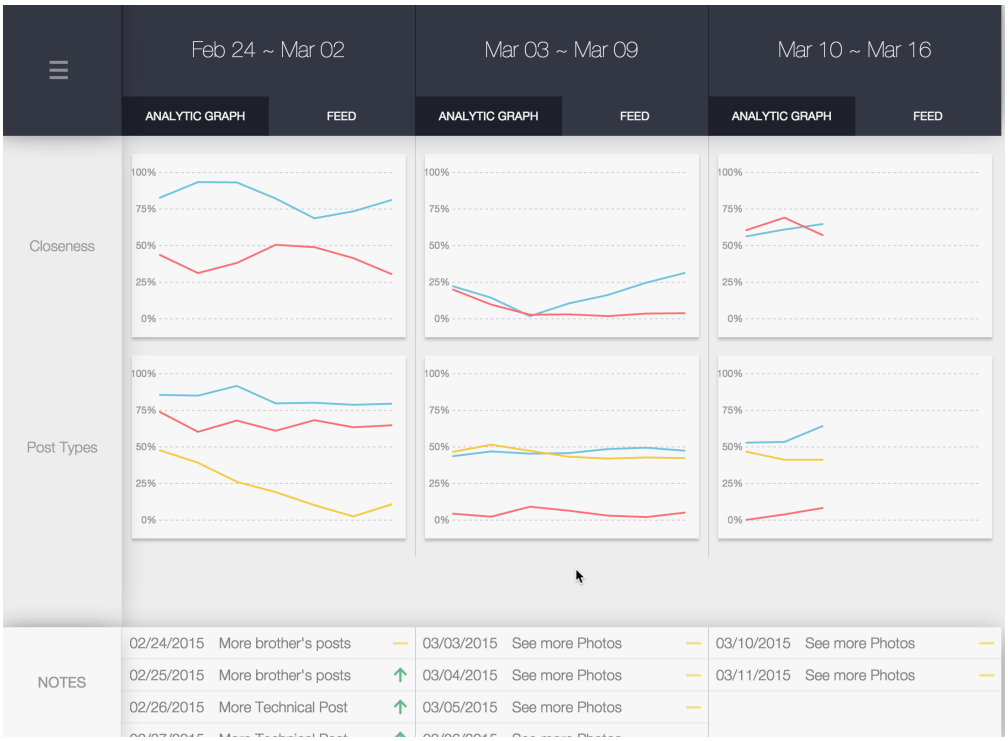
From the design perspective, this interface could be generalized to other feed-based social networks beyond Facebook or Twitter, and other data with features. Because the key concepts of the interface are time, feature and feed. None of them are bounded to a particular platform. Therefore this design could be used for any data with features, with subtle tweak, to compare and explore the trend of features over a certain dimension.

3.10 Curation View Challenges

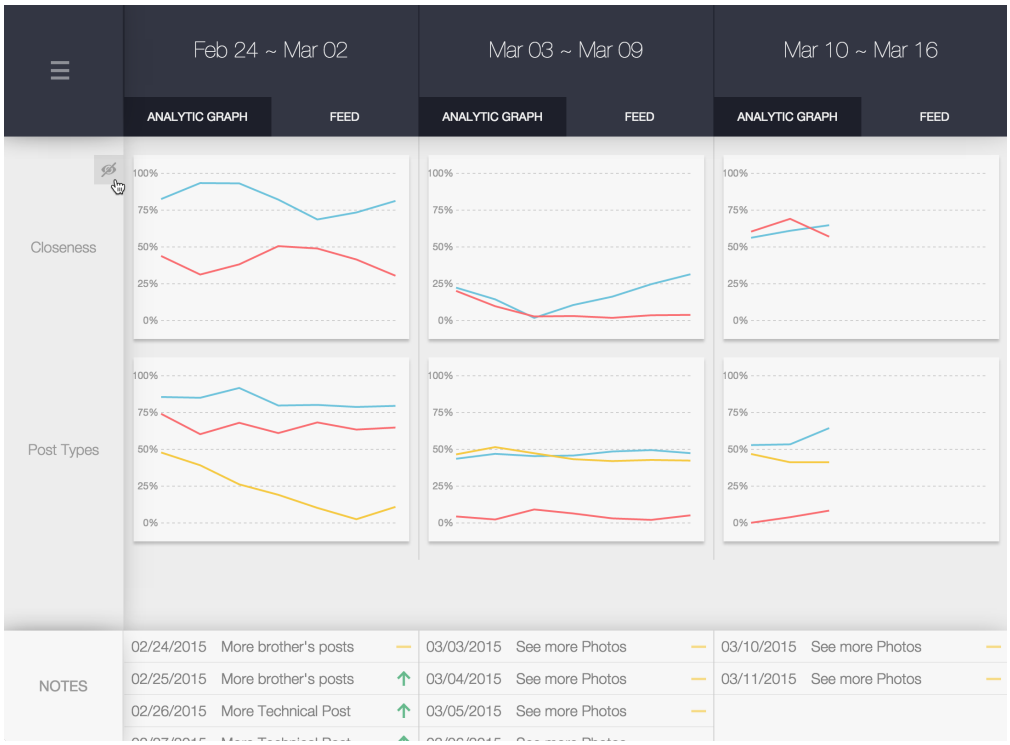
With less interaction, the main purpose of curation view is to display the feeds' contents. As in Fig. 1.2, compared to the other view, interface elements here are minimized to save space for the contents. Also, to help with comparison of curated feeds, we highlight the same posts between the two columns with background color. There is also a histogram to display the proportion of posts in two feeds. The curated list is always using the purple color for consistency. Finally, to make the labeling process straightforward, we use the same icons as in Twitter for our user study.

In summary, Fiddler is designed to fulfill its goal of helping users make sense of curation algorithms. From major features such as layout and scroll, to minor ones like hovering and icon choices, we want to provide users with an encouraging environment to compare and explore the algorithms.

3.11 Figures

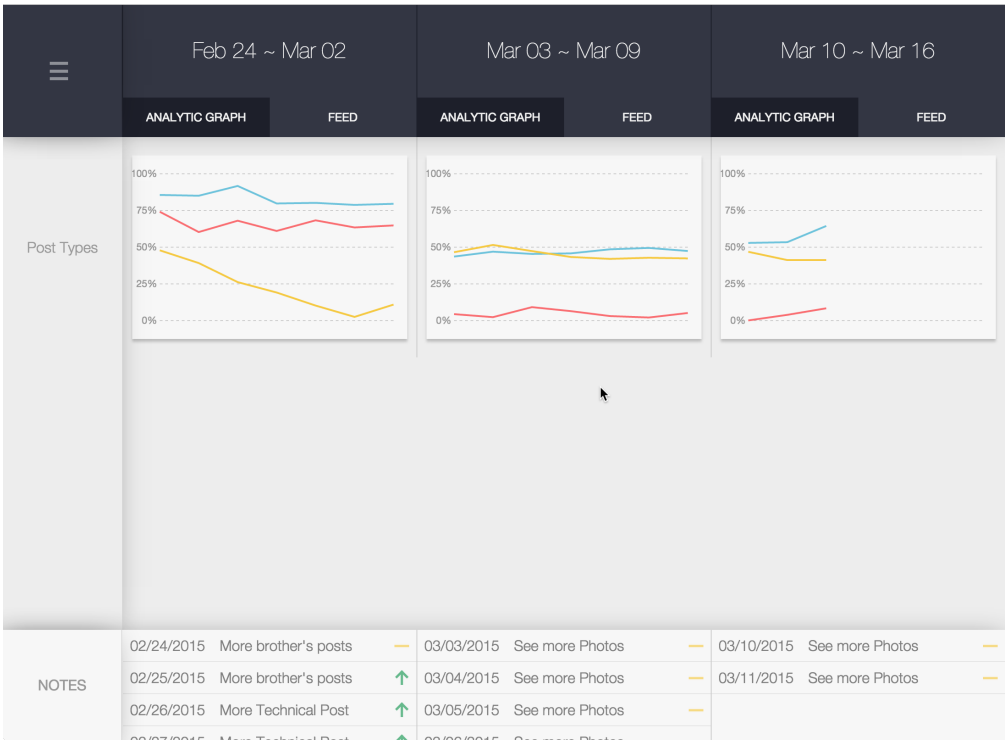


(a) Home page

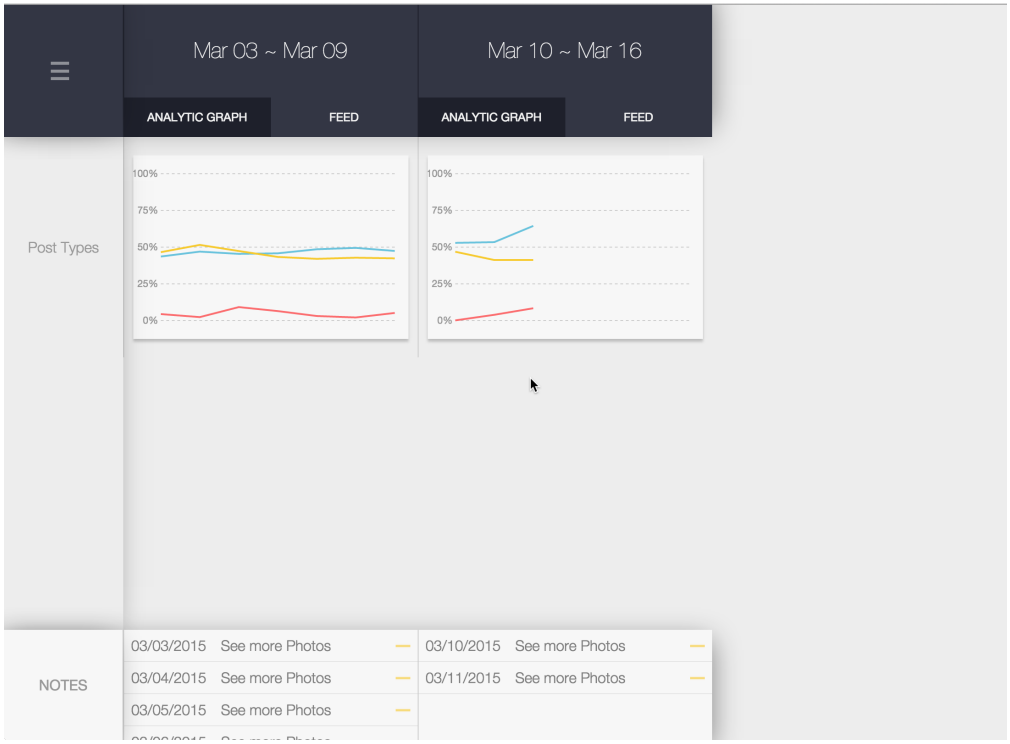


(b) Click hide button

Figure 3.1: Comparison view use case: goal setting

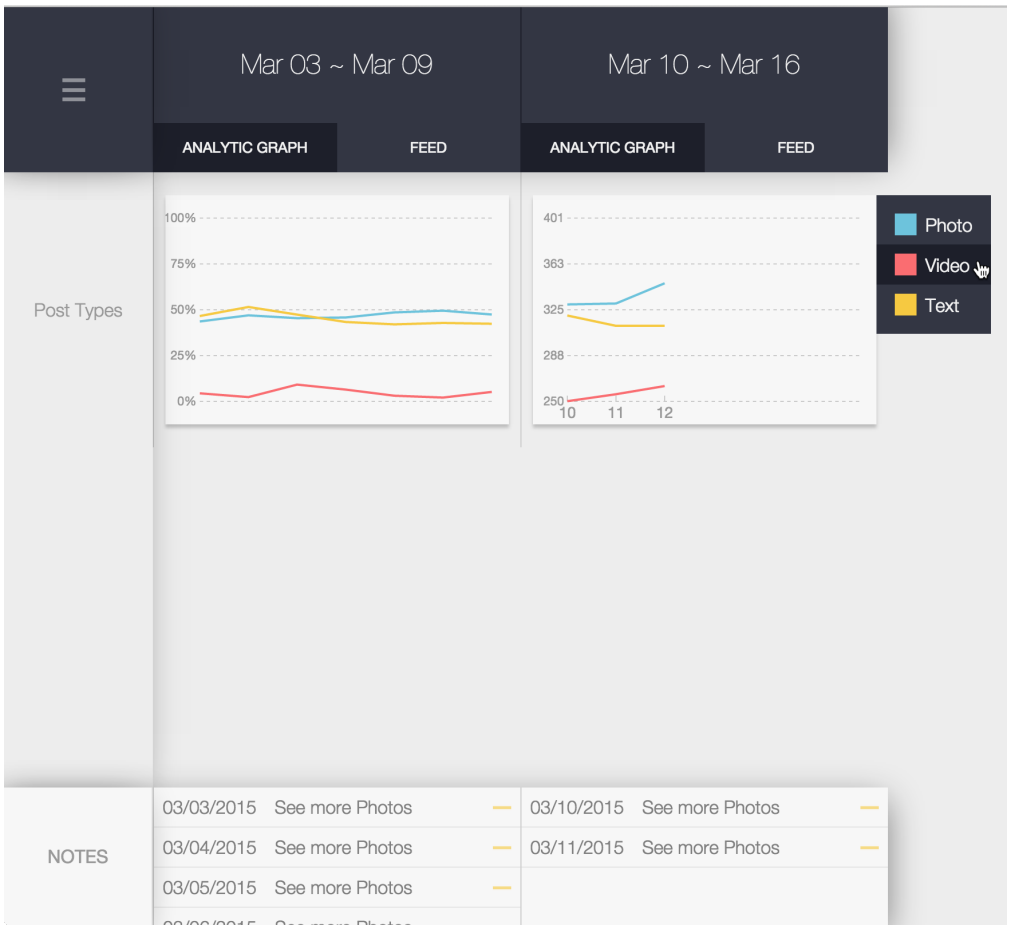


(c) Hide irrelevant row



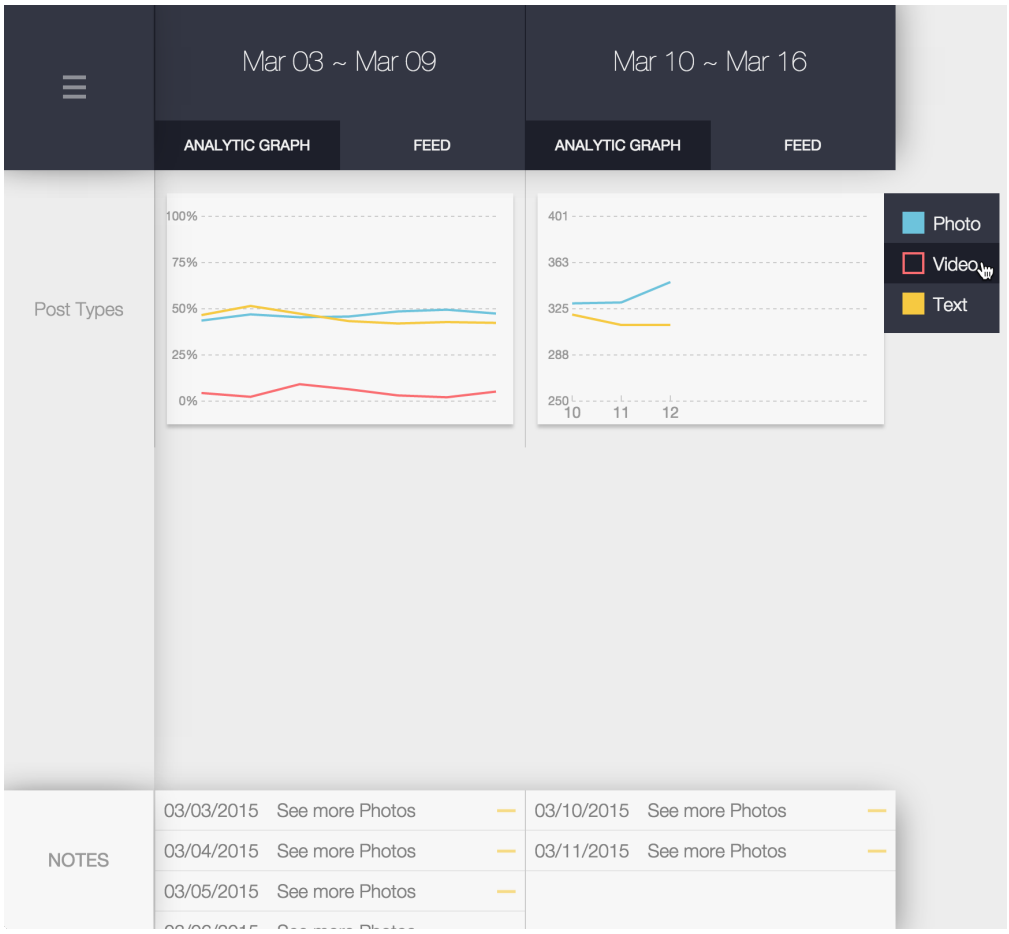
(d) Hide irrelevant column

Figure 3.1: Comparison view use case: goal setting (cont.)



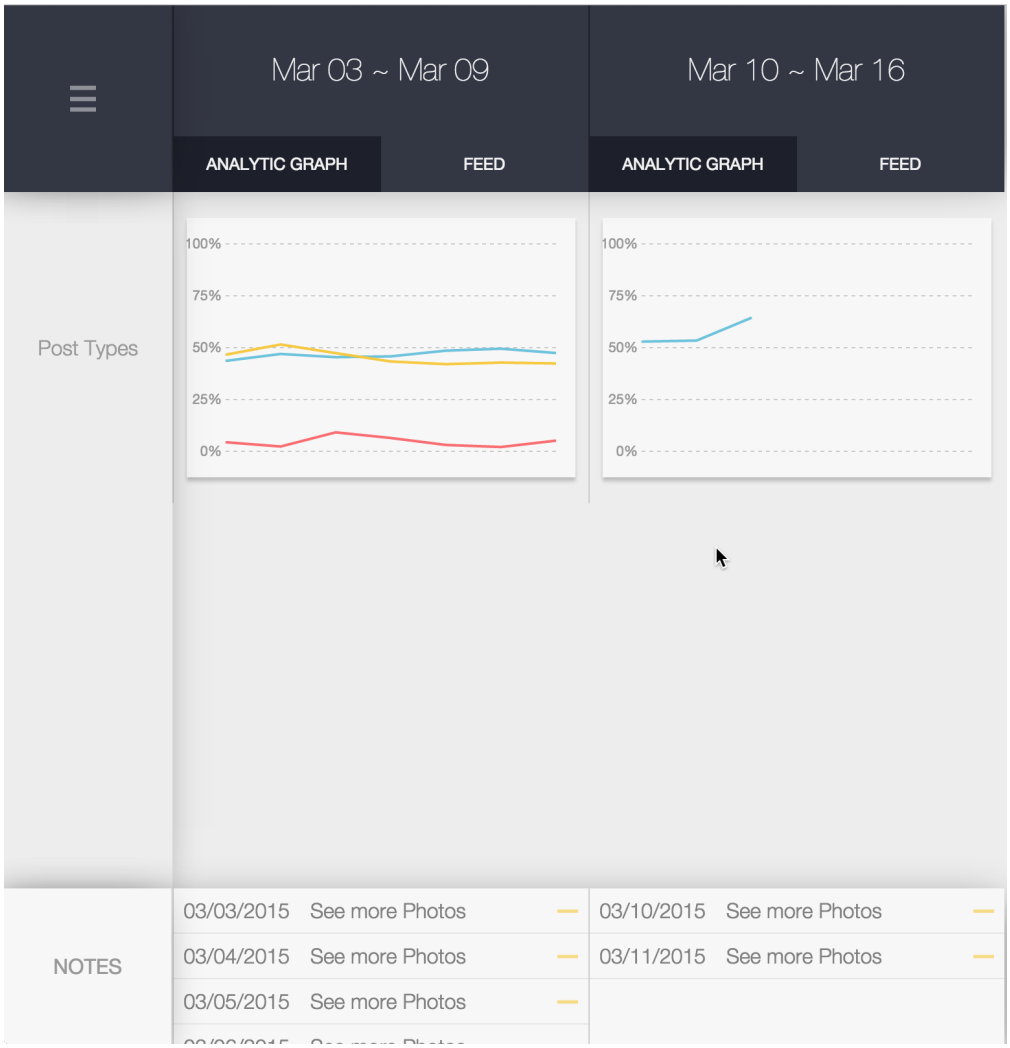
(e) Click legend button

Figure 3.1: Comparison view use case: goal setting (cont.)



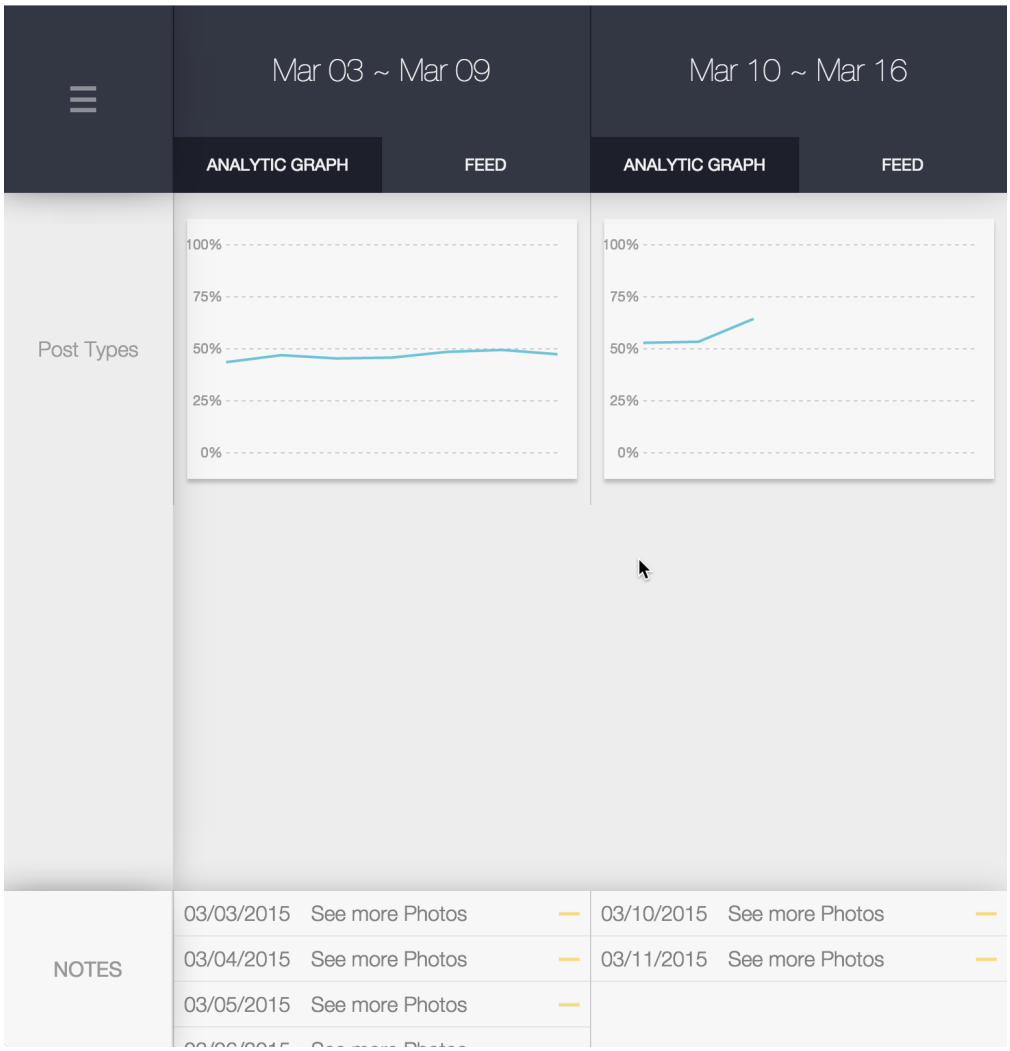
(f) Hide selected series

Figure 3.1: Comparison view use case: goal setting (cont.)



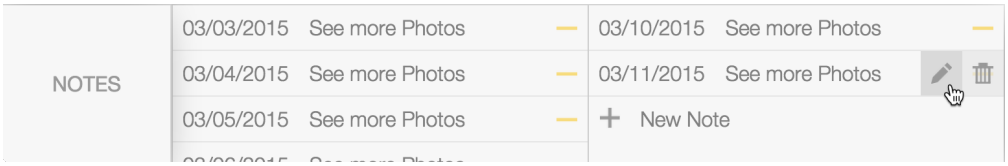
(g) Hide other irrelevant series

Figure 3.1: Comparison view use case: goal setting (cont.)

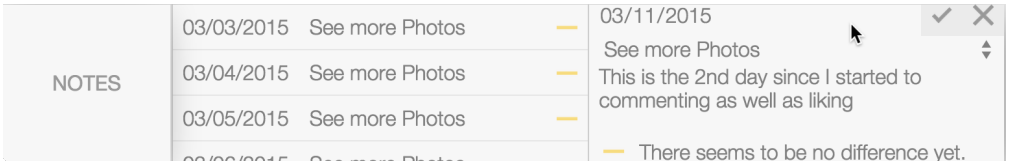


(h) Hide irrelevant series in the other graph

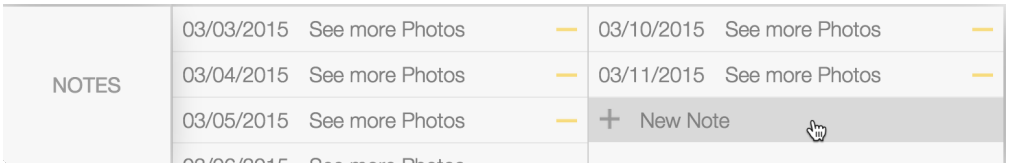
Figure 3.1: Comparison view use case: goal setting (cont.)



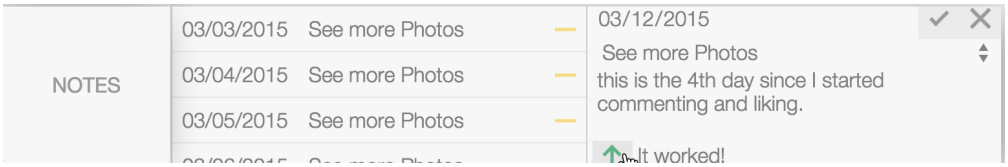
(i) Click to show a previous note



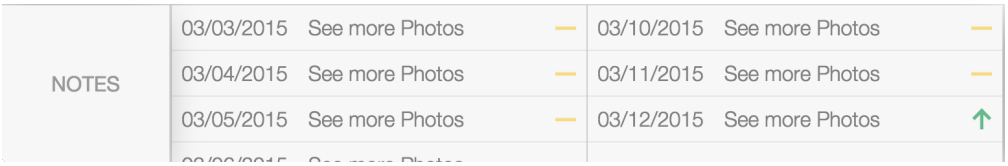
(j) Read the previous note



(k) Create a new note



(l) Fill in the information



(m) Save the note for today

Figure 3.1: Comparison view use case: goal setting (cont.)

(a) Before curation

Figure 3.2: Curation view use case: curating non-friend Tweets

(b) After curation

Figure 3.2: Curation view use case: curating non-friend Tweets (cont.)

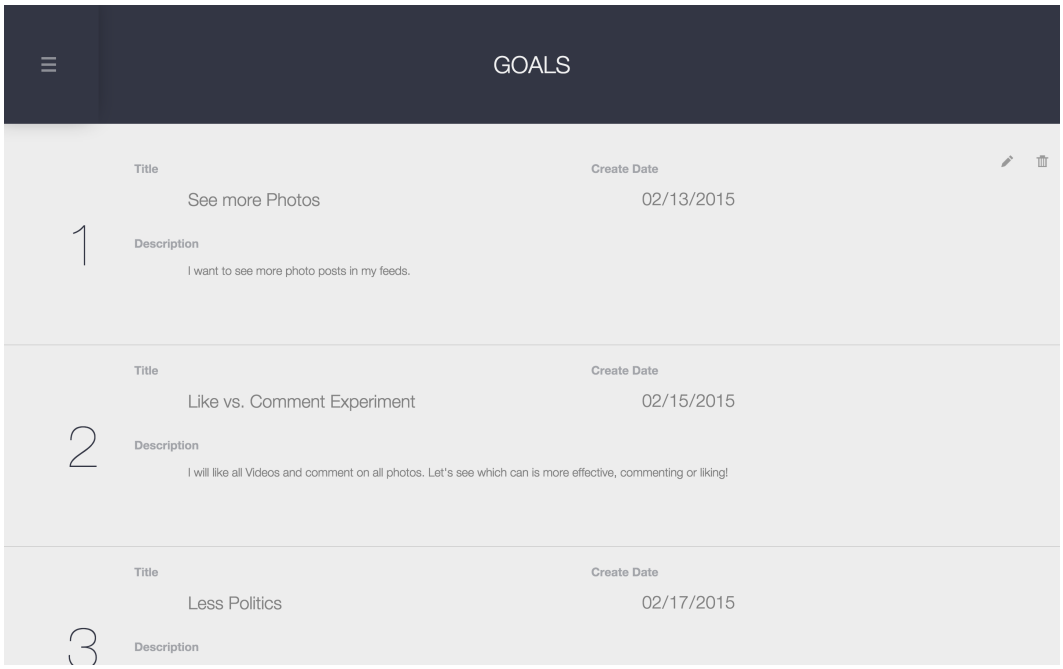


Figure 3.3: Page for setting up goals

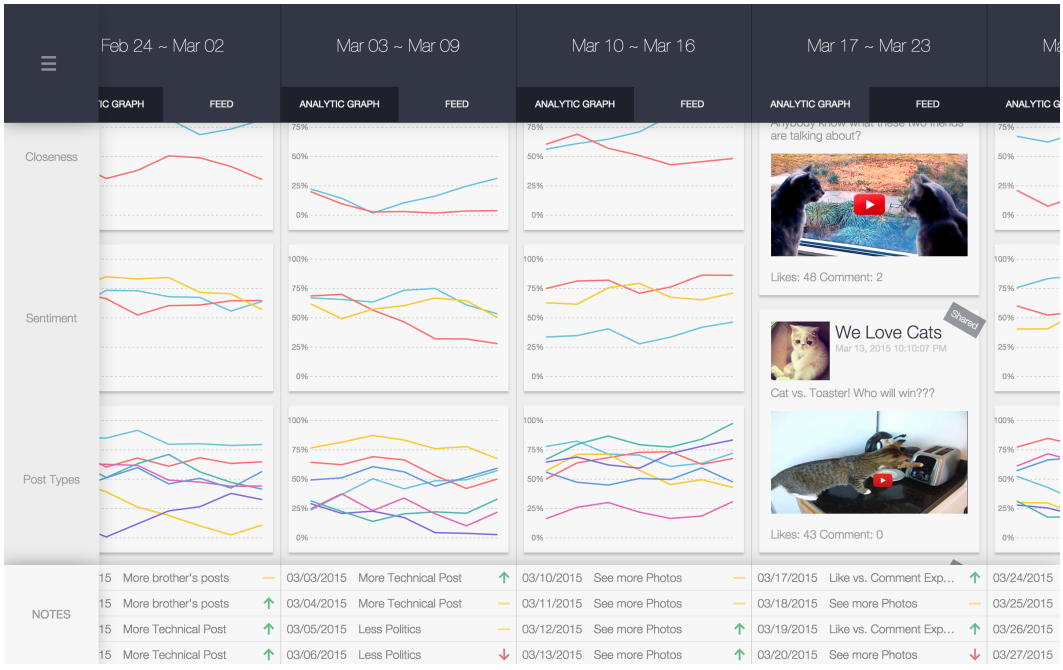


Figure 3.4: Synchronized scrolling and feed contents

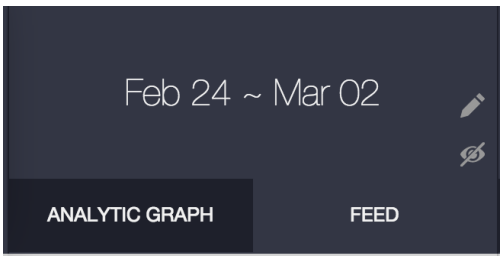


Figure 3.5: Hide and edit column buttons

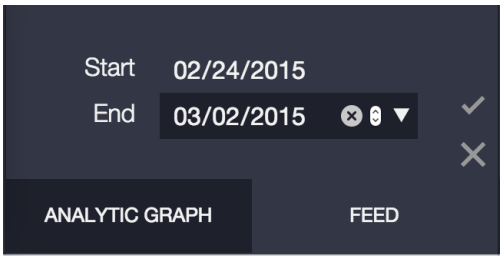


Figure 3.6: Edit time period of column

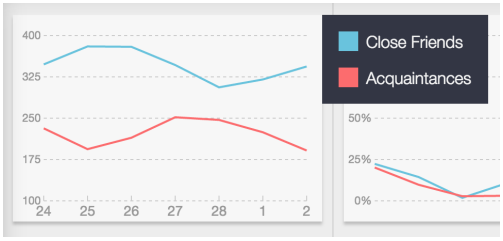


Figure 3.7: Legend indicates both series' colors and their visibility

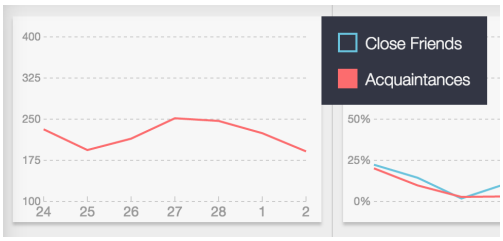


Figure 3.8: Use transparent fill color to indicate "Invisible"

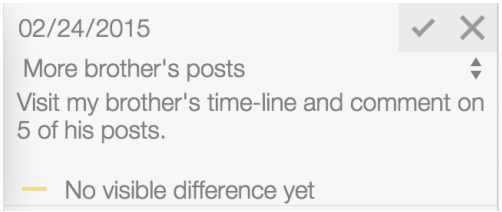


Figure 3.9: Writing a note entry

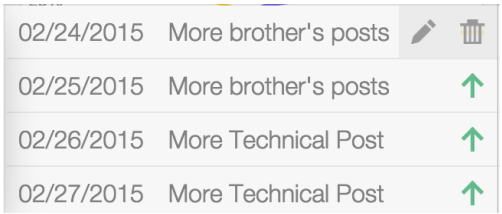


Figure 3.10: Conventional icon choices

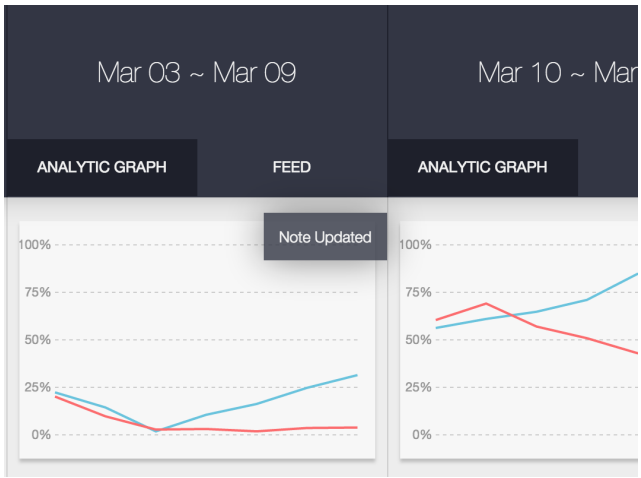


Figure 3.11: Popup message provides confirmation

Chapter 4

Implementation

In this chapter, we will summarize the implementation of the Fiddler system, with a focus on user interface (front-end) besides the system architecture. This chapter begins with the architecture of the entire system, the user interface implementation, and the demonstration of their advantages. Then it introduces the key features in the interface implementation with the related library and language used. Finally, it summarizes the generalization of the system from the implementation perspective. In this section, comparison and curation views are explained together because their implementations are very similar, especially in the key elements discussed below

4.1 System Architecture

There are two sub-systems in the Fiddler system (See Fig. 4.2). The first sub-system consists of the Fiddler server (hereafter "local server") and the social network site (SNS) server, where we retrieve the users' feed. The second sub-system consists of the front-end and the local server, which supports displaying visualization and transferring the user's data.

4.1.1 Fiddler Server and Social Network

The Fiddler server is implemented in PHP and hosted under our research group's domain. It is connected with Facebook using the Facebook Graph API [6] and with Twitter through the Twitter REST API [13]. They both allow us to download user's feeds. At the beginning of the study, we will invite users to log in to our website, which directs them to the SNS to give us access to their feed lists. With their permission, we will download users' feeds once or regularly, as a snapshot of what the users will see at that time point. The downloaded data is processed and saved in our database in both the raw format of feeds and the analysis results, which will be provided to the front-end.

4.1.2 Front-end and Fiddler Server

The front-end (a.k.a. user interface, client-side) and the local (a.k.a. Server-side) are connected through Ajax (Asynchronous JavaScript and XML) [26] requests. When users go to the URL of our website, the front-end files, including HTML, Javascript and CSS files will

be downloaded to their browser and the Javascript code will start running. Then the client-side will send HTTP Requests to the server-side, asking for data to display, such as the trend statistics in the graphs, the users' goals and their notes, which will be returned by the server in JSON media type [21]. Upon receiving the data, the client-side Javascript will generate HTML to display the data. When users save their inputs, these inputs will be sent as JSON to the server to be saved in the database. In summary, the server provides some APIs and the front-end will call these APIs through Ajax to request for users' data. The details of the data requested will be discussed in Section 4.1.4. This subsystem forms a Model-View-Controller(MVC) model [37], where server holds the Model while the front-end holds the View and the Controller.

4.1.3 Advantages of Ajax + API Server architecture

We refer to our structure as Ajax + API Server architecture. Its alternative is the traditional All Server architecture, where all HTML codes are generated on the server-side. There are several advantages of using Ajax + API instead of the traditional architecture.

Firstly, data in JSON format is much smaller than the corresponding HTML to be rendered by the browser. The latter has different HTML elements and many complex structures wrapped around the data, which will considerably increase the size of the data to be transferred. In Fiddler, the situation appears frequently, especially for the graphs that require the analysis of the data, in addition to the raw data, to construct the visualization . Therefore, the website performance will be abated if we choose the All Server architecture, which will in turn degrade the usability.

Secondly, generating HTML on the client-side only changes parts of the web page rather than reloading the entire page. For example, with our current structure, when the user adds one new note in the note list, we will run the client-side code to append the HTML code for that note into the web page. However, with the other option, the server will be informed of the newly added code and regenerate the HTML file for the entire web page, which causes the browser to refresh the entire page. However, because only the note list is affected by the change, a large portion of the HTML code does not need to be regenerated. For Fiddler, it is common for the users to make minor changes to the data, since they can only operate on one goal or note at a time. Therefore, the current architecture generates less duplicated HTML.

Thirdly, as implied by the name, Ajax (Asynchronous Javascript and XML)'s key feature is that all its HTTP Requests are asynchronous. Thus, the web page can operate normally before the server returns the call. Some API calls in Fiddler, such as loading feeds, require a large amount of data to be transferred to the client-side and may take a long time. Because of the asynchronous property, users can remain interacting with the interface while waiting for the data transfer. Without Ajax, the browser will be in a static reload status while the data is loading, and the users will have to wait idly.

Finally regarding development, our current architecture separates client and server-side, linking them only with JSON and APIs. This ensures the modularity of the system. With a commonly agreed protocol on API and data, people could work in parallel to develop both sides, instead of letting the server-side take over the entire MVC model.

4.1.4 API Calls Summary

The APIs are divided into four aspects: graph, feed, note and goals. In general, we kept the APIs as simple as possible. Please refer to Appendix C for a full list of front-end APIs.

The most essential data in this system is the analytic results of feeds, which consists of a four-dimensional array and is used to generate the graphs. We consider the table as a two dimensional array which contains data for each graph. The data of the graph is an array of series, which in turn is an array of data points containing time and values. Also since the notes are based on columns, we put an array of notes in objects that represent columns. Therefore, there are APIs for getting the entire four-dimensional array as well as one single column, which supports adding and editing columns in the interface.

For feeds, we provide an API to simply raw feeds data for display in the interface. However, because of the scale of feeds data, it would cause a performance issue if all feed data that could be displayed were retrieved together. Therefore, we add time period parameters so that the client-side only request feeds for one column when users choose to see them. Also to support pagination, we added start index and count as parameters.

The API for notes supports Create, Read, Update and Delete functions. To simplify the design, we merged Create and Update into one API call because they both have the note object as a parameter. The server returns the note id for client-side, which is used in the Delete function where the entire note object does not need to be transferred. The Read function is already included when reading the graph data. The functions for the Goal APIs are the same as for the notes with the exception of using goal object and a separate Read function to get all the goals for a user.

4.2 Data binding and UI generation with Angular.js

Angular.js [2] is the main library used in the client-side. It is a framework developed by Google based on the MVC model. One of its most powerful features is the two-way data binding, which is widely used in Fiddler.

4.2.1 Two-way Data binding

Data binding is the automatic synchronization of data between the model and the view components. The view is a projection of the model at all times. When the model changes, the

view reflects the change, and vice versa [4]. In the next few sections, we will discuss some key usage of this feature in Fiddler implementation.

4.2.2 Array based UI generation

As mentioned is Section 4.1.4, the graph data received from the server is a four-dimensional array and therefore the UI generated from graph data also has an array-based structure (See Fig. 4.1). With Angular, we created templates of HTML that contains variable that will be filled which values when the actual HTML codes are generated. Then Angular can loop through an array and bind each object to one copy of the template. Therefore, in Fiddler, the main table is generated in nested loops with two levels through each column and row. The graph object is passed to the D3.js [5] to generate the visualizations, which is discussed in Section 4.3. Similarly, the list of notes and goals are generated based on arrays.

Because of the data binding, any changes made to the array variable will be instantly reflected on the UI at all levels of loops. This means reassigning one column object automatically regenerates the entire column by Angular. Similarly, adding a new object in the note array generates new HTML elements for a note with empty fields. This technique saves lots of development efforts because only the model needs to be maintained.

4.2.3 Form Handling

This means reassigning one column object automatically regenerates the entire column by Angular. Similarly, adding a new object in the note array generates new HTML elements for a note with empty fields. This technique saves lots of development efforts because only the model needs to be maintained.

4.2.4 Filtering in binding

This means reassigning one column object automatically regenerates the entire column by Angular. Similarly, adding a new object in the note array generates new HTML elements for a note with empty fields. This technique saves lots of development efforts because only the model needs to be maintained.

4.3 Visualization with D3.js

D3.js [5] is one of the most popular libraries for web-based visualizations, if not *the* most. All the graphs in Fiddler are produced by D3, and we highlight its powerful features specially tuned for visualizations in this section.

D3 also has data binding to view elements. However, its major focus is on SVG elements, which are usually shapes and groups of shapes in the vector image. Different from the templates in Angular, D3 allows developers to bind data to the properties of visual elements in

a descriptive manner through message chaining. With the data binding, we generate visualizations from the graph data, which is also a two-dimensional array (See Section 4.1.4).

Besides having basic shapes, D3 has a great collection of commonly used charts from which we picked the line chart. D3 also supports functions that map values to an axis given the range and the domain of the data. So instead of generating each point or segment, we used these functions to bind our data with axes that were generated automatically.

4.4 CSS Generation with Sass

Sass [12] is a CSS extension language that provides more functions while remaining compatible with CSS. It compiles into CSS files, which makes Sass transparent to browsers. We chose Sass to reduce the unnecessary workload of writing traditional CSS, given the complexity of the Fiddler interface. We list below the benefits we obtained through Sass.

4.4.1 Modularity

Sass allows CSS to be nested, transforming the linear list of repetitive selectors into groups of selectors with a visual hierarchy. Although it produces the same CSS as before, the visual hierarchy offers modularity to the stylesheet. For Fiddler, elements with different parents are grouped separately, which makes it much easier to find and edit the right selector when needed. At the file level, we use the import function, which is similar to the "#include" in C++, so that we could organize styles for different parts of UI in different files.

4.4.2 Dynamic Generation of Styles

Repetitive styles with only minor differences can be commonly found in CSS. In Fiddler, two related problems are the styles for series and the buttons. For the former only the color and class names are different between classes, and for the latter, the picture file names and class names. With Sass, we created a "for" loop that iterates through an array of colors and file names, compiling into a series of selectors with only the differences needed. In this manner, we were able to avoid generating colors in Javascript and also reuse the button class for all buttons, distinguishing them only by class names.

4.4.3 Reuse code with mixin

Mixins are similar to functions in C++ as they take parameters and produce CSS code based on them. They enable parameterized styles. In Fiddler, the styles for shadows appear in multiple selectors, with only offsets being different. Thus, we extracted it into a mixin, with offsets as parameters, and were able to maintain all the shadow effects in one place. Similarly, we extract the button styles to reuse its color, while providing flexibility for hover colors.

The ability to reuse code makes the stylesheets cleaner. For example, all the show and hide styles in Fiddler are produced by mixins so that we can avoid using "display:none", which is not compatible with animation. But we can still using one line to refer to "hide" or "show" by including mixins. For transitions, repetitive attributes are needed for every transition to support different browsers. We used Sass to reuse the styles of transitions, instead of copying them in different places throughout the stylesheet.

4.4.4 Style Profile with viable definition

Finally, because Sass supports variables, we are able to create a list of global variables for color-scheme, font attributes, etc., as a profile for the entire stylesheet. By referencing these global variables in the selector, we maintain consistency of common elements, such as background color and margins, throughout the file. The profile also makes tweaking these styles easier, since they are gathered in one place and reduce the efforts for future improvements, such as updating the color scheme.

4.4.5 Refactoring Stylesheet

Most of the advantages above match with the standard practices in the field of Software Engineering, such as extracting of methods, avoiding hard coded values, use of constants, and so on. With Sass, we can refactor the stylesheet of Fiddler to make it more extensible for future development.

4.5 Generalization from implementation perspective

This implementation is also generalizable due to its modularity and independency from Facebook and Twitter platforms. Using different APIs, the local server can retrieve and analyze data from other websites. The front-end only needs JSON formatted data that contain graph data as arrays to display the visualization. The feeds are merely one example of an array of objects that could be rendered at the front-end. Therefore, this implementation could be used for other SNS or other data with features changing along one dimension. And due to the ease of generalization, we only need to switch to Twitter API and format the output, when implementing the curation view after finishing comparison view.

The implementation of fiddler was aimed for both user experiences and future extension. With the supports from optimized architecture and libraries, we were able to construct Fiddler efficiently and made it easy for future developers to build on.

4.6 Figures

Figure 4.1: Array-Based User Interface Structure

Figure 4.2: Fiddler System Architecture

Chapter 5

User Study

In this section, we will introduce the user study we conducted with Fiddler's curation view. We will start with reviewing the research questions we are trying to answer and explain the procedures to answer them. Then we will present the results along with our answers to the research questions.

5.1 Goals and Procedures

To answer the three research questions on our study, we conducted this user study with Fiddler's curation view and invited each user to compare all Tweets with the non-friend Tweets in their Twitter Home timeline (hereafter "*the timeline*"). We have introduced the questions in Section 1.2. They are listed here again.

RQ1: How aware are users of the curation algorithm in their Twitter Home timeline? What information can help to improve user's algorithm awareness?

RQ2: How satisfied are users with the non-friend Tweets in their timeline? What are the commonalities in users' desired Tweets, if any?

RQ3: How do users report they may modify future behavior after seeing this comparison?

With the first question, we wanted to understand users' awareness of the timeline's curation algorithm, which displayed all the Retweets from users' friends regardless of their authorship. The non-friend Tweets are more likely to be ignored by users, because they are from accounts that users are not following, while having almost the same appearances in the Twitter's user interface. Therefore, before directing users to Twitter, we started the study with a scenario of non-friend Tweet: "Imagine there is a Tweet in your Home timeline and you do not remember you have followed its author." Then we asked what could lead this situation: Did they just forget they followed the author, or there were other reasons? In addition, we listed the latest 200 Tweets in their timeline with all the non-friend Tweets in the curation view. With the Retweet information hidden, we let users guess why the non-friend Tweets appeared in their timelines. These two steps can reflect users' awareness of the curation algorithm and also we can understand how to improve algorithm awareness from the cues that lead to their conclusions.

With the second question, we wanted to know whether users were satisfied with non-friend Tweets in their timelines. Because when users choose to follow someone, they could be subscribing only to that user's original Tweets. Compared to original Tweets from users' friends, non-friend Tweets could be less satisfying, because users did not directly choose to receive them. To study this question, we first asked user to give a subjective rating with 7-point Likert scale on the relevancy of their timelines. Then in the curation view, we let users labeled each non-friend Tweet to indicate whether they wanted it to appear in their timelines and whether they wanted to follow its author. From these results, we could see the satisfaction of users with the Tweets from someone they did not follow. The labeled Tweets showed users' desired Tweets over their timelines. And we were able to find the commonalities among the Tweets they liked to see.

For the last question, we were asking whether there would be influences on user's future behavior from the comparison of curation algorithms. So we discussed with users about their future interactions with Twitter after seeing the comparison between non-friend and all Tweets on their timeline. With these findings, we could know the possible changes of behavior on social network sites after users used Fiddler to compare curation on their feeds.

Please refer to Appendix A for the pre-interview survey and Appendix B for the original scripts used in our study.

5.2 Subjects

To reflect Twitter users' opinions and behavior, we recruited five students at University of Illinois at Urbana-Champaign as participants (four men, one woman), whose ages were between 24 and 30 years of age. All of them were frequent Twitter users, following at least 100 users. Three of them were following 250 users. All participants used Twitter at least 2 times during the week before our study and used it regularly on daily basis.

5.3 Results

5.3.1 Users' Algorithm Awareness (RQ1)

For the non-friend Tweet scenario, two users suspected it could be a Retweet. Three users thought they might forget about following the imaginary author. And three users mentioned the Tweet could be an advertisement or suggested Tweet. Two users answered with multiple possibilities for this scenario. As mentioned in Section 1.2, while there are reports that algorithmically selected "suggested" Tweets appear in one's timeline [40], we did not observe any "suggested" Tweets in our study. So the non-friend Tweets in the participants' timelines were all Retweets of their friends at the time of our study. As we can see, without looking at the feeds, only two users mentioned the possibility of Retweet.

When users examined the non-friend Tweets in Fiddler, they picked 39 Tweets in total to guess why those Tweets appeared on their timelines (See Table 5.1). We asked each of the five participants to pick 10 Tweets. They (P1, P2, P3, P4, P5) picked 5, 6, 9, 2, 11 Tweets respectively. This resulted in a total of 39 Tweets. We recorded only two Tweets from P4, because the participant gave a general description after picking two Tweets: *"I think those are [about] trending topics. My friends ... or other news sources are also into this news trend. They shared it. I think that's why [these Tweets] get into my home timeline."* (P4)From the 39 selected Tweets, our participants guessed that 33 of them were Retweets Among the 39 Tweets, users guessed that 33 of them were Retweets. We also counted the number of Tweets guessed to be Retweet for each individual user and compared it with the number of all guesses made by the user (See Table 5.2). As we can see, for each user, at least 65% of the user's guesses were Retweets. For the other Tweets they did not pick, users also tended to assume they are Retweets. *"Some of these I don't have a lot of context, so I assume someone [I'm following] retweeted them."* (P1) From these results, we can see, when seeing a Tweet not from their friends, our participants mostly knew the Tweet would be a Retweet, rather than suggested ones or advertisements. This showed their awareness of Twitter's curation algorithm, which displayed Retweets from users' friends regardless of their authorship. The differences between the guesses made before and after seeing the curation comparison also showed the presence of Tweets and comparison helped users to understand why the Tweets appeared in users' timelines.

When looking at those non-friend tweets, all users started guessing with suspecting they did not follow the authors. Then they explained why they thought a Tweet could be retweeted by their friends. With open-coding method, we categorized their reasons into the following types:

1. The participant had followed other users who were related to the general topic of the Tweet, for example one user followed NASA and suspected a space-related Tweet is retweeted by NASA.

2. The participant had followed other users who were related to the author of the Tweet. For example, *"this right here is a program that I know somebody else I followed runs. So I imagine they are Tweeting from their other account."* (P2)

3. The participant had followed other users who were related to specific contents mentioned in the Tweet. For example, *"I assume I have a friend . . . who retweeted it, because he has a work at [the name of a company]"* (P5)

4. The participant had followed other users who appeared in the Tweet, such as in the "@" tags or photos.

5. The Tweet was about a popular topic. For example, *"those are trend topics and (I think) a lot of people are retweeting it."* (P4)

The results are summarized in Table 5.3. As we can see, users are judging based on the Tweets' relationships to their friends. To conclude a Tweet was retweeted by friends, they mostly judged by the Tweet's social network relation, topic and content relevancy to their friends on Twitter. To answer this research question, according to our results, participants were able to tell most of the non-friend Tweets are Retweets and therefore were aware of Twitter's curation algorithm. The Tweet's relevancy to users' social circle on the SNS helped users to make their judgments.

5.3.2 User's satisfaction with non-friend Tweets(RQ2)

With "7" for very relevant, three users gave the subjective rating of "5" for the relevancy of their timelines. One gave *"4 or 5"* (P2) and the other user *"7"* (P3). Tweets were regarded as irrelevant if they have topics not interesting to users or they are about friends' personal lives or have repeated information.

Table 5.4 shows the percentage of non-friend Tweets users liked to see on their timeline. Although users gave positive subjective rating for timelines, 4 users chose to see less that 35% of non-friend Tweets. As for following, users explained they did not easily follow others, but rather started by checking other's personal timeline. The user (P3), who gave a subjective rating of"7", picked only two non-friend Tweets to see. *"Usually the people I followed are content creators and when they Tweet about the content they created, ...I found cool. But when they retweet ...it's usually like an ad. And feels like it's not that interesting."* (P3)

And from the first row in Table 5.5 we can see, non-friend Tweets actually took over 26% to 47% of users' feeds. Therefore, even if we assume users were willing to see all their friends' original Tweets, there were still 16% to 41% of unwanted Tweets in their timelines. These numbers showed that most of our participants were not interested in most of the non-friend Tweets.

We also used open-coding method to summarize users' reasons why they liked a Tweets (See Table 5.6). Users randomly picked and explained 44 Tweets they liked in total. Interesting, controversial topics and entertaining contents were the main reasons users liked Tweets. Only one user liked Tweets regarding friends' professional life for networking. And another user liked one Tweet about friends' personal lives. We can see personal lives were the least mentioned reasons. We suspect users may be using Twitter as an information source for knowledge or entertainment, but rather than a way to connect with their personal friends. This result showed a topic or entertainment based curation could be ideal to user.

We furthered this discussion by asking users what was their ideal curation over their timelines. Two users suggested to filter out some Tweets based on topics. *"If the news that I follow don't overlap at all with that particular topic, then it seems that topic is probably not going to be something I'm interested in."* (P5) Three users suggested curate based on their actions on Twitter, such as who they followed or what they favorited. Two users wanted to have more

feedback with the curation algorithm. *"I would probably try to get more feed back, like favorite stuff or retweet stuff. just so that I get the algorithm to know what I liked."* (P2) However, users were concerned that they may miss out contents if the curation algorithm is *"too strict"* (P2). For this concern, one user wanted to keep using the current curation algorithm on Twitter, to see all original Tweets and Retweets from the user's friends. *"I would leave (the algorithm) always shows me everything. As long as it's not like I'm missing on content, . . . it should be good to me."* (P3) From their thoughts, we can learn that most users were open to filter out some Tweets they currently see. But they wanted to have control over what they want to see.

In conclusion for this question, most users wanted 27% of non-friend Tweets on average in their timeline. And most of their desired Tweets had interesting or entertaining topics.

5.3.3 User's future behavior (RQ3)

At the end of the study, we asked users if there would be any changes of their interaction with Twitter. Four users said they would pay more attention on non-friend Tweets and unfollow those who posted contents not interesting. *"Just looking through this, it makes me want to go through and curate some of my users away."* (P2) In fact, users said they were already using unfollow or mute to curate what they see. The comparison gave them more insight on the non-friend Tweets retweeted by users' friends. *"Not everyone is retweeting, I can see quite a few people over and over just retweeting stuff into my feed. Maybe I'd rather get rid of them, than see all their noise."* (P2)

In summary, through this user study, we found out users were aware of the curation algorithm in the current Twitter Home timeline. Users knew the non-friend Tweets were Retweets from their friends. But they only wanted to have 27% non-friend Tweets in their timelines. We learned that users' desired Tweets usually had interesting or controversial topics or entertaining contents. With insight from the comparison, users reported they would pay more attention to non-friend Tweets and curate their timelines by unfollowing people. With more curation algorithms supported in Fiddler, we hope users can compare different curated feeds and explore their personal curation in the future.

5.4 Tables

Guessed Answer	Number of Tweets	Percentage
Retweet	33	84.62%
Advertisement	3	7.69%
Not sure	3	7.69%
Total	39	100%

Table 5.1: Participants' answers to why non-friend Tweets appear in their timelines

Participant Id	P1	P2	P3	P4	P5	Mean
Tweets guessed as Retweets	83.33%	66.67%	100.00%	100.00%	84.62%	84.62%

Table 5.2: Percentage of Tweets participants assumed (guessed) were Retweets.

Reasons	Total
1. Related to general topics	8
2. Related to the author of the Tweet	8
3. Related to specific contents	6
4. Appear in Tweets	5
5. Related to Popular topics	3
Total	30

Table 5.3: Participantsfb reasons for assuming a Tweet was a Retweet (Please refer to Section 5.3.1 for full description of the reasons)

Participant Id	1	2	3	4	5	Mean
Liked	34.67%	24.05%	5.77%	54.79%	13.83%	27.08%
Followed	8.00%	8.86%	0.00%	8.22%	0.00%	5.09%

Table 5.4: Percentage of Liked or Followed Tweets among non-friend Tweets in each participant's timeline

Participant Id	1	2	3	4	5	Mean
Non-friend Tweet	38.27%	41.15%	26.67%	37.24%	47.72%	38.22%
Unwanted Tweet	25.00%	31.25%	25.13%	16.84%	41.12%	27.87%

Table 5.5: Percentage of non-friend Tweets and unwanted Tweets among all Tweets in each participant's timeline

Concept	Sub-Concept	Occurrences
Topics	Interesting	22
	Controversial	6
	Total	28
Entertainment	Funny Tweets	4
	Nice Picture	5
	Total	9
Social Circle	Professional life	6
	Personal life	1
	Total	7

Table 5.6: Reasons participants were happy to receive a non-Friend Tweet

Chapter 6

Conclusion

In conclusion, there are two major contributions of this study. Firstly, the Fiddler system was built with design considerations and extensible implementation, to help user compare between time periods and curated feeds with the comparison and curation views, and eventually help them make sense of curation algorithms. The system is generalizable across different social networks or data in the similar format. Secondly, a user study was conducted with curation view for the non-friend Tweets on Twitter. The study found frequent users were aware of the curation algorithm. The relationships between the Tweets and users' social networks helped them make judgments. This result is different from the FeedVis study [24], where 62.5% of participants were not aware of the curation algorithm on Facebook News Feed. The difference could be caused by the fact that Twitter's curation algorithm had a simpler model than the one on Facebook. In addition, our participants were all frequent Twitter users. They may have active engagement with Twitter, which could lead to users' awareness according to the FeedVis study.Regarding users' satisfaction on non-friend Tweets, users wanted to see 27% of non-friend Tweets on average, among which they were interested in those related to their interests and those were controversial or entertaining. Most users reported they would curate their timeline by unfollowing their connections on Twitter after this study. With the study using Fiddler comparison view in the future, we will be able to know what are users' goals and explorations regarding curation algorithms. Also to extend the study with curation view, more curation algorithms will be added for comparison. With comparison and curation views, we hope Fiddler will continue to help users make sense of newsfeeds.

References

[1] Amazon Mechanical Turk. https://www.mturk.com.

[2] Angular JS. https://angularjs.org.

[3] Credit Karma. https://www.creditkarma.com/.

[4] Data Binding. https://docs.angularjs.org/guide/databinding.

[5] Data-Driven Documents. http://d3js.org.

[6] Facebook Graph API. https://developers.facebook.com/docs/graph-api.

[7] FAQs about following. https://support.twitter.com/articles/14019-faqs-about-following.

[8] FAQs about Trends on Twitter. https://support.twitter.com/articles/101125-faqs-about-trends-on-twitter.

[9] How News Feed Works. https://www.facebook.com/help/327131014036297/.

[10] Klout. https://klout.com/.

[11] Material Design. http://www.google.com/design/spec/material-design/introduction.html.

[12] Sass: Syntactically Awesome Style Sheets. http://sass-lang.com.

[13] Twitter REST API. https://dev.twitter.com/rest/public.

[14] What is News Feed. https://www.facebook.com/help/210346402339221.

[15] XRay. http://xray.cs.columbia.edu.

[16] R. Baldwin. Twitter is testing showing you tweets from accounts your friends follow. http://thenextweb.com/twitter/2014/08/04/twitter-testing-showing-users-tweets-accounts-friends-follow/.

[17] N. Cawthon and A. V. Moere. The effect of aesthetic on the usability of data visualization. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 637–648. IEEE.

[18] P.-H. Chiu, G. Y.-M. Kao, and C.-C. Lo. Personalized blog content recommender system for mobile phone users. *International Journal of Human-Computer Studies*, 68(8):496–507, 2010.

[19] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, and Z. Popovi. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[20] C. D. Correa and K.-L. Ma. *Visualizing social networks*, pages 307–326. Springer, 2011.

[21] D. Crockford. The application/json media type for javascript object notation (json). 2006.

[22] E. Design. Why we love (or hate) everyday things. *New York: Basic*, pages 21–38, 2004.

[23] N. Diakopoulos. Algorithmic accountability reporting: On the investigation of black boxes. *Tow Center for Digital Journalism, Columbia University*, 2014.

[24] M. Eslami, A. Rickman, K. Vaccaro, A. Aleyasen, A. Vuong, K. Karahalios, K. Hamilton, and C. Sandvig. I always assumed that I wasnt really that close to [her]: Reasoning about invisible algorithms in the news feed. CHI, 2015.

[25] C. GARLING. Tricking Facebook's Algorithm. `http://www.theatlantic.com/technology/archive/2014/08/tricking-facebooks-algorithm/375801/`.

[26] J. J. Garrett. Ajax: A new approach to web applications. 2005.

[27] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[28] S. Grissom, M. F. McNally, and T. Naps. Algorithm visualization in CS education: comparing levels of student engagement. In *Proceedings of the 2003 ACM symposium on Software visualization*, pages 87–94. ACM.

[29] T. Halleck. Facebook Hacks: 9 Tips And Tricks To Get Your Posts On Top Of The Newsfeed. `http://www.ibtimes.com/facebook-hacks-9-tips-tricks-get-your-posts-top-newsfeed-1709170`.

[30] J. Heer and D. Boyd. Vizster: Visualizing online social networks. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 32–39. IEEE.

[31] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, 1952.

[32] D. Holmes. Why you shouldnt freak out about Twitters new timeline experiments. `http://pando.com/2014/10/17/why-you-shouldnt-freak-out-about-twitters-new-timeline-experiments/`.

[33] M. Honan. I Liked Everything I Saw on Facebook for Two Days. Heres What It Did to Me. `http://www.wired.com/2014/08/i-liked-everything-i-saw-on-facebook-for-two-days-heres-what-it-did-to-me/`.

[34] L. Hong, G. Convertino, B. Suh, E. H. Chi, and S. Kairam. FeedWinnower: layering structures over collections of information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 947–950. ACM.

[35] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[36] M. Ingram. Should You Care How High Your Klout Score Is? `http://www.bloomberg.com/bw/technology/should-you-care-how-high-your-klout-score-is-10272011.html`.

[37] G. E. Krasner and S. T. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, 1(3):26–49, 1988.

[38] T. Lavie, M. Sela, I. Oppenheim, O. Inbar, and J. Meyer. User attitudes towards news content personalization. *International journal of human-computer studies*, 68(8):483–495, 2010.

[39] T. Munzner, F. Guimbretire, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: scalable tree comparison using Focus+ Context with guaranteed visibility. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 453–462. ACM.

[40] T. O'Brien. The spirit of experimentation and the evolution of your home timeline. `https://blog.twitter.com/2014/the-spirit-of-experimentation-and-the-evolution-of-your-home-timeline`.

[41] E. Rader and R. Gray. Understanding User Beliefs About Algorithmic Curation in the Facebook News Feed.

[42] H. Rappaport. Messing With My Credit Scores: An Experiment, Part I. `http://www.frugaltravelguy.com/2015/02/messing-with-my-credit-scores-an-experiment-part-i.html`.

[43] D. Rodrigues and I. Oakley. *Social Circles: A 3D User Interface for Facebook*, pages 838–839. Springer, 2009.

[44] J. Russel. Twitters latest experiment turns favorites into retweets (and its annoying lots of people). `http://thenextweb.com/twitter/2014/08/17/twitters-latest-experiment-turns-favorites-into-retweets-and-its-annoying-lots-of-people/`.

[45] Saintnicks. Gaming the system: how changes to Twitter's algorithm could be considered harmful. `http://saintnicks.uk.com/gaming-the-system-how-changes-to-twitters-algorithm-could-be-considered-harmful/`.

[46] R. R. Sambasivan, I. Shafer, M. L. Mazurek, and G. R. Ganger. Visualizing request-flow comparison to aid performance diagnosis in distributed systems. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2466–2475, 2013.

[47] J. W. T. M. S. W. C. K. M. B. B. A. Smith. Can an Algorithm Know the Real You?: Understanding Peoples Reactions to Hyper-personal Analytics Systems.

[48] C.-Y. Tseng, Y.-J. Chen, and M.-S. Chen. Socfeedviewer: A novel visualization technique for social news feeds summarization on social network services. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 616–617. IEEE.

[49] Z. Tufekci. Why Twitter Should Not Algorithmically Curate the Timeline. https://medium.com/message/the-algorithm-giveth-but-it-also-taketh-b7efad92bc1f.

[50] A. Vande Moere, M. Tomitsch, C. Wimmer, B. Christoph, and T. Grechenig. Evaluating the effect of style in information visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12):2739–2748, 2012.

[51] V. Woollaston. Justin Bieber fans beat Twitter 'block'. http://www.webuser.co.uk/news/top-stories/492646/justin-bieber-fans-beat-twitter-block.

[52] J. Yarow. Here's The Magic Word That Can Boost Your Post In Facebook's News Feed. http://www.businessinsider.com/facebook-news-feed-ranking-2014-4.

Appendix A

Pre-Interview Survey

1. What is your gender?
    - Male
    - Female
2. What is your age range?
    - 18˜25
    - 25˜35
    - 35˜50
    - 50˜64
3. What is your occupation/major?
    - Would you describe yourself as
    - American Indian/Native American
    - Asian
    - Black/African American
    - Hispanic/Latino
    - White/Caucasian
    - Pacific Islander
    - Other
4. What is your total household income annually?
    - Less than $10,000
    - $10,000 to $19,999
    - $20,000 to $29,999
    - $30,000 to $39,999
    - $40,000 to $49,999

- $50,000 to $59,999
- $60,000 to $69,999
- $70,000 to $79,999
- $80,000 to $89,999
- $90,000 to $99,999
- $100,000 to $149,999
- $150,000 or more

5. Which one(s) of the following social media do you regularly use?

    - Facebook
    - Twitter
    - Pinterest
    - Quora
    - Reddit
    - LinkedIn
    - Email
    - Skype
    - Others:

6. Among these social media, which one(s) are your favorite(s)?
7. How many years have you used your favorite social media?
8. Did you use Twitter yesterday? If so, how many times?
9. How many times did you use Twitter in the last week?
10. How would you describe your use of Twitter on a typical day? (When do you typically log into Twitter?)
11. What information is the most valuable to you on Twitter? In other words, what do you think you'd miss if you left Twitter?

    - Shared Links (URLs)
    - Photographs
    - News from other users
    - People's life events (birthdays, weddings, …)
    - Status updates

- Just-in-time meetings (“Hi I’m at …, anyone want to join me?”)
- Direct messaging
- Lists
- Trending Topics
- World news stories
- Sport stories
- Technology stories
- Science stories
- Design stories
- Searching for people
- Searching for topics
- Others:

12. What content do you tend to ignore or not use on Twitter?

- Shared Links (URLs)
- Photographs
- News from other users
- People’s life events (birthdays, weddings, …)
- Status updates
- Just-in-time meetings (“Hi I’m at …, anyone want to join me?”)
- Direct messaging
- Lists
- Trending Topics
- World news stories
- Sport stories
- Technology stories
- Science stories
- Design stories
- Searching for people
- Searching for topics
- Others:

Appendix B

Interview Questions

1. Imagine you see a tweet on your Home timeline, but you don't remember you have followed its author. Do you think this is because you forgot you have followed this person or there are other reasons that lead to this situation?
2. *Ask participant to go to Twitter account. Write down the users name, number of tweets, following, followers and register Date*
3. What's the first thing you typically check or read when you log in to Twitter?
    - *If participant do not know what terms to use* There are some key components of the Twitter website. The webpage you see when you go to twitter.com is your Home timeline, which contains posts from other users. The trending topics are those hashtags listed on the side of this page in a section named "Trending topics". The page you see after you go to click a user's name is that person's personal timeline. Twitter Lists are lists of users, which you can use to view tweets from users contained in a list. The Discover page contains a list of tweets suggested to you by Twitter.
4. While you are logged in at Twitter, what approaches do you use to read other users' tweets?
    - Go through the Home timeline
    - Search by hashtag/trending topics/other topics,
    - Visit the personal timelines of those you are following.
    - Go through your Following List
    - Looking at Discover pages
    - Other approaches *ask what is the approach*
5. Does your Home timeline provide useful and relevant information for you? Could you give a rate from 1 to 7, 1 being not at all, 7 being strongly relevant?
    - *If less than 7* what are the information in your Home timeline, do you think is not relevant?

6. What actions do you take to modify what you see in your Home timeline?

    - *If participant does not give an answer* For example, have you choose to block or mute people, have you authorized twitter to suggest tweets based on your website visits, etc.

7. *Show All vs. non-Friend Tweets Comparison* (The RT tags, Graphs and the percentage of non-friend tweets are hidden)

8. In this interface, you can see two columns. Each contains a list of tweets. Each tweet has the author's username, tweet time, tweet content. The "All" column contains the latest 200 tweet in your Home timeline we get with the Twitter API. The "Subset" column contains a subset of tweets we select from the "All" column. In both columns, if a tweet is colored in purple, it means it has been selected into the "Subset" column.

9. *Show the "star" and "follow" button*

10. Please look at the "Subset" column, where each tweet has a Star and Follow button. Please read each tweet in this column, then click the Star button, if you actually want this tweet to appear in the Home timeline, and click the Follow button, if you want to follow the author of this post. This operation will not make any change to your twitter account. Could you pick 10 tweets that you have liked or followed and explain why you like or follow it?

11. *After user finish, Hide the "star" and "follow" button*

12. Please look at the from the "Subset" column. Please pick about 10 tweets from this column and explain your thoughts about why each of them appear on your Home timeline.

13. *Show the RT tags and graphs*

14. The posts in the "Subset" column are all the posts that are in your Home timeline but created by someone you are not following. As you can see, some tweets are marked with a tag on the top-left corner. Tweets with these tags are retweets from the users you are following. The name of the user who make a retweet is indicated in the tag. Tweets without such tag are originally created by someone you are following.

15. As we discussed before all the tweets in the Subset column are not originally created by someone you are following. And it turns out all these tweets come from the retweets of someone you are following.

16. The graph is a histogram based on the tweets from the two columns. The X axis has hours in the past few days and the Y axis is the amount of tweets in each time period.

The purple bars are corresponding to the Subset column. And as we discussed, the Subset column are tweets not originally created by those you are following. For example, this means from … to …, there are … posts in total and … of them are not created someone you follow.

17. What is your expectation regarding the amount of tweets that is created by someone you didn't follow. Does these results meets your expectation?

18. Do you wish Twitter use a curation algorithm that selects what tweets to show you?

    - *if yes* Why? And what content do you want them to select or eliminate for you?
    - *if no* Why?

19. Do you think the knowledge from this study might influence how you use Twitter in the future? How would you change your interaction with Twitter?

Appendix C

Front-end APIs

- Graphs
  - `getAllGraphs(userId)`

    Explanation Get all the data for all the graphs.

    Return A JSON object of all graph data in all date range

  - `getColumnData(userId, start_timestamp, end_timestamp)`

    Explanation When the user change the date range for a column, this method is called to update the graph data in the column.

    Return A JSON object of graph data in a column.

- Feeds
  - `getFeeds(userId, start_timestamp, end_timestamp, start_index, count)`

    Explanation Get all the feeds a certain range from a certain index. Suppose the feeds are in an array A, then feeds objects from `A[start_index]` to `A[start_index - count - 1]` should be returned.

    Input The start and end timestamp for the date range. The index of the first feed and the number of feeds needed.

    Return A JSON array of feeds as requested

- Notes
  - `addorUpdateNote(userId, note_obj)`

    Explanation If the `note_odj` does not contain an Id, it is a new note entry to be added, otherwise or update existing note entry.

    Return `note_id`. `note_id` is assigned by backend. if the query is success, Return note id else Return -1.

  - `removeNote(userId, note_id)`

    Explanation Remove note from database.

    Return A Boolean value, indicating whether the note in removed.

- Goals
  - `addOrUpdateGoal(userId, goal_obj)`

    Explanation If the `goal_odj` does not contain an Id, it is a new goal entry to be added, otherwise or update existing goal entry.

    Return `goal_id`. `goal_id` is assigned by backend.
  - `getGoals(userId)`

    Explanation This methods is used to get the user's goal list when the web page is loaded.

    Return A JSON array of all goals.
  - `removeGoal(userId, goal_id)`

    Explanation Remove goal from database.

    Return A Boolean value, indicating whether the goal in removed.